Uncertainty-aware Causal Graph Learning

LI-CHENG YEH

School of Computer Science
The University of Auckland

Supervisor: Meng-Fen Chiang, Wang-Chien Lee

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Professional Studies in Master of Data Science, The University of Auckland,2023.

Abstract

Graph Neural Networks (GNNs) have gained widespread adoption for learning representations of graph-structured data and have been applied to various tasks, including node classification and graph classification. Despite their success in capturing intricate structures in graph data, studies have revealed that graph encoders like GNNs are prone to severe biases. This sensitivity to biases can result in accurate predictions based on spurious correlations. To mitigate this issue, the debias graph encoder has recently been developed and integrated into GNN architectures. It aims to identify and reduce biases during information propagation and aggregation. However, existing approaches to uncover the underlying rationale and provide explanations for prediction outcomes have proven unsatisfactory, necessitating further improvement. Motivated by this, in this study, we formulate the problem of graph rationale learning, which aims to uncover the reasons behind graph prediction outcomes by utilizing estimated uncertainty in the disentangled rationale and its counterpart. To achieve this, we propose the Uncertainty-Aware Causal Graph (UACG), a novel debiasing mechanism for GNN-based graph encoders. UACG combines rationale learning techniques with GNNs to identify the true causal subgraph and offer interpretability for explaining prediction outcomes, as well as understanding the biases introduced by GNNs without a debiasing mechanism. Specifically, our approach incorporates an uncertainty estimation mechanism that categorizes nodes into high or low uncertainty and employs sparse regularization to improve generalization. This allows our method to generate rationales that emphasize the most relevant graphical features while disregarding irrelevant or redundant ones. Extensive experiments demonstrate that our proposed UACG framework effectively addresses biases in GNNs by identifying the causal subgraph in biased datasets of varying degrees and discovering rationales that accurately predict graph labels. Overall, UACG enhances the explainability of rationale and the accuracy of GNN-based graph classification tasks by incorporating interpretability and uncertainty-aware estimation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, Graph Neural Networks (GNNs) have emerged as an impressive framework for analyzing and learning from complex graph-structured data. GNNs have gained immense popularity due to their ability to capture intricate relationships and dependencies in such data. They perform well in various applications, including graph classification tasks, which are prevalent in domains such as biology [12], social networks [12], and knowledge graphs [45]. In most domains, GNNs have achieved significant advancements that cannot be ignored. Their ability to leverage the graph structure has revolutionized the field of graph-based machine learning.

However, despite their remarkable capabilities, GNNs also face limitations in graph classification tasks. One major challenge is their vulnerability to biases present in the underlying data. Due to the message-passing property in GNNs [5], biases can influence the training process in various ways, including biased node representations, skewed neighborhood aggregation, or overemphasizing certain subgraphs. Such biases can lead to unfair or inaccurate predictions. Therefore, it becomes crucial to develop mechanisms that effectively debias GNNs during the learning process. The debias encoder is a module within the GNN architecture explicitly designed to identify and reduce biases during information propagation and aggregation. By incorporating a debias encoder, GNNs can learn to discriminate between relevant and biased information, enabling fairer and more reliable predictions. This encoder enhances the model's ability to distinguish genuine relationships from biased or spurious patterns, thereby improving the overall fairness and accuracy of GNNs.

1.1 Research Motivation

GNNs ideally utilize subgraphs to predict labels when the graphs are unbiased [37], indicating that only causal subgraphs are related to the graph labels. However, graph datasets inevitably contain biases due to uncontrollable data collection methods. Consider Figure 1.1 where most of

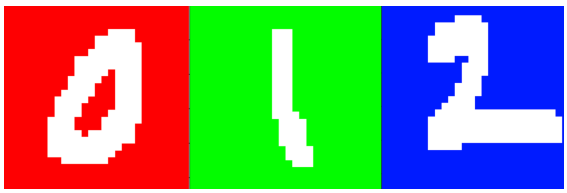


Figure 1.1: CMNIST-75sp Dataset [2].

the digit backgrounds are the same, such as the red background for “zero”. In this case, GNNs do not need to learn the correct function to achieve high accuracy in predicting the digit labels. Instead, it is much easier for them to learn statistical shortcuts that link the background color with the most frequently occurring digit in each case. Unfortunately, such methods generalize poorly when encountering out-of-distribution (OOD) data, where the background color changes in the testing dataset. Consequently, these non-causal subgraphs, which are only marginally related to the causal part, provide limited exposure to the causal subgraphs for predicting the labels.

Motivated this, recent research on rational learning [51, 8, 43, 46] aims to uncover the reasons behind machine learning models’ predictions. By explicitly modeling and reasoning about the decision-making process, rationale learning provides a foundation for understanding and controlling biases in the model’s predictions. This approach introduces interpretability, enhancing confidence in the model’s output. Moreover, in all datasets, both causal and non-causal parts always coexist, regardless of their syntactic or real-world nature. Under this condition, the causal and non-causal subgraphs identified by biased GNN models are highly uncertain. To address this issue, I introduce an uncertainty-aware estimation method inspired by [36, 20]. This estimation approach retains the model’s knowledge on specific high-uncertainty nodes while compensating for low-uncertainty nodes through sparse regularization [1]. Sparse regularization is a technique used to promote sparsity in the learned representations of nodes or edges in a graph. The ultimate goal of sparse regularization is to identify and emphasize the most relevant features or connections while disregarding irrelevant or redundant ones.

Due to the message passing nature of GNNs, variants such as GCN and GAT primarily perform well on homogeneous graph data. However, it is crucial to handle the heterogeneous nature of graph datasets carefully, as different types of nodes or edges in the neighborhood may have varying influences on the target node’s representations, potentially disrupting the aggregated representation for the target node. Several studies have made attempts to address the heterogeneous nature of graph data [18][61][19][59]. For instance, Xiao Wang et al. [49] proposed HAN (Heterogeneous Graph Attention Network) to capture and aggregate information from different types of nodes and edges in a heterogeneous graph. Chuxu Zhang et al. [59] proposed a framework that handles both structural and feature heterogeneity in a graph by employing type-specific aggregation and update functions. Ziniu Hu et al. [19] extended the

Transformer architecture to handle heterogeneous graphs. These approaches aim to enhance the modeling capabilities of GNNs on heterogeneous graph data. Nevertheless, the consideration of label conformity in causal graph learning has not been explored in the context of heterogeneous graph data.

To the best of our knowledge, no existing research has specifically addressed the generation of causal graphs by incorporating label conformity and uncertainty of sampled graphs in a unsupervised manner to achieve attentive parameter learning. The closest related study is the research by Yongduo Sui et al. (CAL) [46], which formulates the task of generating attentions to disentangle causal and non-causal features in the input graph. Similarly, Shaohua Fan et al. (2022) propose DisC (Disentangled Causal substructure) [43], which employs a parameterized edge mask generator to explicitly separate the input graph into causal and biased subgraphs. While these models demonstrate the interpretability and generalization capabilities of GNN classifiers compared to conventional variants, they make the assumption that the sampled graphs have equal importance in adjusting the graph encoder parameters. Unfortunately, this assumption may lead the graph encoders to capture causal graphs that deviate from the ground-truth causal graphs to some extent. In conclusion, there is no existing research that addresses the gap in evaluating the contribution of each sampled graph from the perspective of certainty in causal graph learning in a unsupervised manner.

1.2 Research Contribution

To bridge this gap, we propose the Uncertainty-Aware Causal Graph (UACG), a debiasing mechanism designed for GNN-based graph encoders. The UACG model utilizes subgraph sampling and estimation techniques to facilitate rationale learning. This approach guides the model to focus on sampled subgraphs that contribute higher certainty to the explanation of prediction outcomes. Specifically, we design an uncertainty estimator for sampled graphs by evaluating their contribution to accurately explaining the end-task prediction outcome, such as in the case of graph classification tasks. Given a graph and its ground-truth label, we formulate graph rationale learning as the task of generating disentangled causal and non-causal graphs. To achieve this, we first learning the causal attention graph and the non-causal attention graph from the label conformity perspective. To sample graphs, we generate L-hop subgraphs for each node in the disentangled causal and non-causal subgraphs obtained. To assess the contribution of each L-hop subgraph, we propose an uncertainty estimator that categorizes each L-hop based on its confidence in contributing to the final explanation for the end-task prediction. Consequently, the subgraphs with higher confidence have a greater impact on shaping the parameters of the graph encoder, while the subgraphs with lower confidence have a smaller impact on reshaping the encoder parameters. We demonstrate that training the model with both high and low confidence sampled graphs enhances the robustness of the graph encoder against potential biases and spurious correlations in capturing the end-task. Extensive experiments demonstrate that UACG

can effectively explain a graph's classification label with fidelity across four benchmark datasets, including two synthetic and two real-world datasets. In the ablation study, removing the label conformity component leads to an accuracy decline to 62.3%, while removing the uncertainty-aware component results in a decline to 65% compared to our full model (74.89%). These findings demonstrate the potential efficacy of the module in guiding the generation of causal graphs. Lastly, we provide qualitative visualizations to demonstrate the causal graphs learned by UACG compared to other strong baselines. The following summarises the primary contributions of the thesis:

- We propose the Uncertainty-Aware Causal Graph (UACG), a debiasing mechanism designed for GNN-based graph encoders. UACG aims to disentangle rationale and non-rationale information from the input graph to provide an explanation for a prediction outcome.
- To ensure accurate aggregation at an early stage, we propose a label conformity estimator to identify homogeneous and heterogeneous edges in the graph.
- To facilitate attentive parameter learning, we propose an uncertainty estimator that differentiates the contributions of sampled graphs. This enables the optimization of graph encoder parameters with varying impacts based on their importance as determined by the uncertainty estimator.
- We define a joint learning objective that incorporates disentangled attention learning, label conformity estimation, and uncertainty estimation for sampled graphs in contributing to the explanation of end-task predictions.
- We conduct quantitative and qualitative studies on four benchmark datasets to address five research questions. UACG outperforms existing methods in terms of end-task performance and rationale quality, providing superior explainability for the prediction outcomes.

1.3 Structure of Thesis

The thesis is structured as follows.

- Chapter 2 - Literature Review: This chapter provides a comprehensive review of existing literature in the research domain. It encompasses relevant works in the fields of graph neural networks, rationale learning, and sparse regularization. Furthermore, it explores the application of sparse regularization techniques to enhance rationale explainability. Additionally, the chapter investigates biases in current graph neural network models and their impact on various graph-related tasks.

- Chapter 3 - Preliminary: This chapter provides an overview of key concepts relevant to the study. It covers discussions on Graph Encoders, which are integral to the research, the state-of-the-art mechanism for Debiasing Graph Encoders, as well as the chapter explores Graph Rationale Learning. Finally, the chapter formally defines the problem of causal graph learning.

- Chapter 4 - Methodology: This Chapter presents the framework of the study, the Uncertainty-Aware Causal Graph (UACG) framework. The UACG framework aims to generate high-quality causal graphs that provide rationales for target end-task prediction labels. A key aspect of the framework is the incorporation of an uncertainty estimator, which allows sampled graphs with varying levels of uncertainty to influence the parametrized GNNs differently.

- Chapter 5 - Experiment: This chapter presents the quantitative and qualitative experimental results for five research questions using four benchmark datasets, which include both synthetic and real-world datasets.

- Chapter 6 - Conclusions: This Chapter highlights the achievements of the current study, discusses the limitations observed, and suggests potential directions for future research.

Chapter 2

Literature review

In this chapter, we review the literature in the field of graph neural networks, rationale learning, and sparse regularization. In Section 2.1, we discuss the characteristics of graph neural networks and the types of tasks they can solve. In Section 2.2, we introduce the key components of GNNs, namely the graph encoder, and discuss the different types of graph encoders available. In Section 2.3, we explore the latest research progress on rationale learning and its application to GNNs. In Section 2.5, we examine the need for label conformity mechanisms in graph encoders and discuss the current research approaches to address this problem.

2.1 Graph Neural Networks (GNNs)

Deep learning has emerged as a key strategy in artificial intelligence. Traditional deep learning techniques, such as recurrent neural networks [42] and convolutional neural networks [25], have been highly successful in processing Euclidean data, such as images and sequential data like text. However, many real-world phenomena and scientific problems are naturally expressed or better explained using complex systems, such as diagrams or objects. Examples include social networks, recommendation systems, drug discovery, and systems analysis. Graph-structured data is used to represent such complex systems and contains rich structural and semantic information that can encode complex patterns. This type of data does not exhibit as much spatial locality and structure as image or text data, making it unsuitable for highly regularized neural structures like convolutional neural networks and recurrent neural networks. As a result, there has been a significant increase in interest within the research community regarding deep learning on graphs [11, 52].

Among the approaches developed for deep learning on graphs, Graph Neural Networks (GNNs) have emerged as highly successful learning algorithms for various tasks in multiple application areas. Architectures such as Graph Convolutional Networks (GCN) [22], Graph Attention Networks (GAT) [48], and Graph Isomorphism Networks (GIN) [53] have demonstrated

remarkable performance in well-established domains like social networks and bioinformatics. Furthermore, these architectures have found applications in other scientific research fields, including recommendation systems [12], computer vision [54], and natural language processing [9].

GNNs are specialized neural architectures designed to process data structured as graphs. The objective of GNNs is to iteratively refine node representations by combining the representations of neighboring nodes with their own representation from the previous iteration. GNNs can perform three main graph analytics tasks. The first task is node-level analysis, where the outputs of GNNs are associated with tasks involving node regression and classification. This task is considered the most intuitive and basic in GNNs. The primary goal of GNNs is to learn representations of nodes based on neighboring nodes, and node classification involves classifying nodes based on these learned representations. Once each layer of the GNN model generates node-level representations, GNNs can perform graph-level analytics tasks. This task requires adding graph pooling layers to compute graph-level representations based on these node representations. Graph-level representations capture key knowledge learned from the input graph and are crucial for graph classification. Additionally, edge-level analytics tasks, which involve predicting missing or future connections between pairs of nodes, have long been a focus in GNN research. However, in this study, we will focus on graph classification.

Despite the achievements in existing research, GNNs still face several challenges when applied to highly structured data that is time-evolving, multi-relational, and multi-modal. Furthermore, since GNNs learn from neighbors, it is inevitable to introduce biased information during the learning process.

GNNs can be trained using supervised, semi-supervised, and unsupervised learning methods, but each of these methods depends on different graph analytics tasks. In semi-supervised learning for node classification, as shown in Figure 2.1, the graph encoder learns the hidden representation of each node by aggregating the features of its neighbors. Following the feature aggregation operation, a non-linear transformation is applied to generate the output. This framework enables GNNs to learn a robust model for classifying unlabeled nodes in the node classification task.

For edge prediction tasks, GNNs utilize a decoder that calculates the pairwise distance between nodes in an unsupervised manner. On the other hand, supervised learning is primarily employed for graph-level tasks. Figure 2.2 presents the simplified graph classification framework, which is the focal point of our study. In contrast to node classification, GNNs in this task introduce a pooling layer to summarize the node representations and generate the graph representations. Once the graph representation is learned, an MLP (Multi-Layer Perceptron) and Softmax are utilized to predict the graph label.

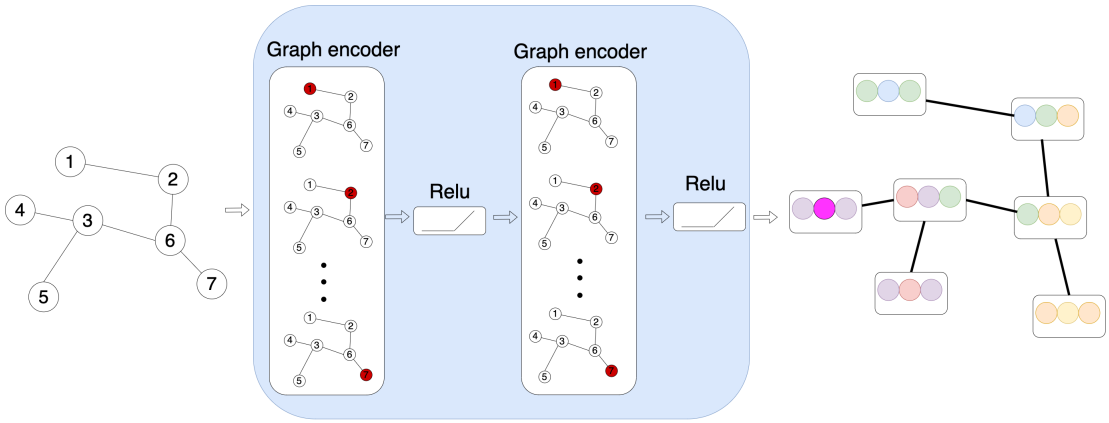


Figure 2.1: Node Representation Learning Framework.

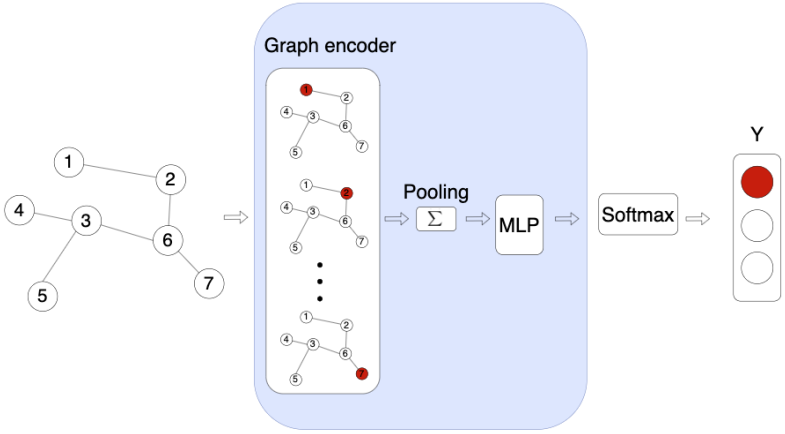


Figure 2.2: Graph Classification Framework.

2.2 Graph Encoders

Graph encoders are the most important component of GNNs, which mainly aggregate information from neighboring nodes to generate the node representation.

2.2.1 Convolutional Neural Networks (CNNs)

Before delving into the key aspect of GNNs, let us review some concepts of CNNs [25]. In traditional CNNs, the convolution operation involves a filter mechanism that acts as a sliding window across the entire image, enabling CNNs to learn features from neighboring cells. Spectral-based GNNs and spatial-based GNNs, which will be discussed later, were inspired by this mechanism.

2.2.2 Spectral-based Approaches

Spectral-based approaches have been widely used in GNNs for graph representation learning. These approaches, also known as graph signal processing GNNs, view the node features as signals and employ the Fourier transform for signal conversion. In the Fourier domain, the convolutional operation is transformed into the multiplication between the graph signal and the filter. This type of graph convolutional operation can be interpreted as noise removal from graph signals [44].I have selected four papers that have made significant contributions to spectral-based approaches in GNNs

Firstly Spectral CNN [6] by Bruna et al. (2014). This paper investigates the effectiveness of CNNs in image and audio recognition tasks, emphasizing their ability to leverage local translational invariance within signal data. The authors explore adaptations of CNNs for signals defined on more diverse domains that lack a translation group. They propose two approaches: one based on the hierarchical clustering of the domain and another based on the spectrum of the Laplacian graph. The paper demonstrates the feasibility of training convolutional layers for low-dimensional graphs, laying the foundation for subsequent spectral-based GNNs. Although this paper does not design a dedicated GNN framework, it proves the feasibility of convolutional layers on graph data, so all Spectral-based GNNs are derived based on it. Graph convolutional network(GCN) [22] by Kipf and Welling (2017). This paper introduces a scalable method for semi-supervised learning on graph-structured data using an efficient variant of convolutional neural networks specifically designed for graphs. The authors justify their choice of the convolutional architecture by employing a localized first-order approximation of spectral graph convolutions. The proposed GCN framework achieves linear scalability with respect to the number of graph edges and learns hidden layer representations that capture both local graph structure and node features. Both Spectral CNN and GCN are highly representative methods in spectral-based approaches, with GCN being a more dedicated framework for graphs.Cayleynets [30] by Levie et al. (2018). This paper presents a novel convolutional architecture in the

spectral domain for GNNs. The key component of Cayleynets is a parametric rational complex function called Cayley polynomials, enabling efficient computation of spectral filters that focus on specific frequency bands of interest. This approach addresses the limitations of GCN, which may not be suitable for real-world data exhibiting varying sizes and connectivities. AGCNs [31] by Li et al. (2018).This paper introduces a learnable adaptive parameter in the graph Fourier transform to adjust the spectral response of the graph signal adaptively. This adaptive graph convolutional operation allows the model to learn task-specific adaptations for each input graph data during the training process.

2.2.3 Spatial-based Approaches

In contrast to spectral-based approaches that rely on a solid mathematical foundation to construct graph signals for learning node representations, spatial-based approaches define graph convolutions intuitively based on a node's spatial relations. I have selected six papers that have made significant contributions to spatial-based approaches in GNNs. Firstly MPNNs [15]by Gilmer et al. (2017), introduced Message Passing Neural Networks (MPNNs) for predicting properties of molecules. MPNNs use a recursive message-passing scheme, where each node updates its representation based on the representations of its neighbors. GraphSAGE [17] by Hamilton et al. (2017), which focuses on learning node representations in large-scale graphs. GraphSAGE utilizes a trainable aggregator function to combine information from neighboring nodes, enabling effective representation learning. DiffPool [56] by Ying et al. (2018) , which proposes a differentiable pooling mechanism to learn hierarchical representations of graphs. The authors introduce a pooling function that dynamically selects the most informative nodes in a graph based on their importance, facilitating the learning of meaningful hierarchical representations. ClusterGCN [10] by Chiang et al. (2019), which presents a clustering approach to reduce the computational complexity of training deep and large graph convolutional networks. The authors propose a two-stage training procedure involving clustering the nodes in a graph and training a subgraph with fewer nodes. Deep Graph Convolutional Neural Network(DGCNN) [60] by Zhang et al. (2019). DGCNN introduces a localized graph convolution model that demonstrates connections with two graph kernels. It facilitates the extraction of relevant features from the graph and enables the organization of the graph's vertices in a consistent order, making it possible to apply traditional neural networks to graph data. Finally, two of the most representative spatial-based graph encoders are Graph Attention Network (GAT) [48] by Petar Veličković et al. (2018) and Graph Isomorphism Network (GIN) [53] by Keyulu Xu et al. (2019). These encoders address the limitations of GNNs in distinguishing simple graph structures. GIN proposes a permutation-invariant function to learn node representations in a graph, allowing GINs to learn invariant representations of graphs under node permutations. On the other hand, GAT is a novel GNN method that is different from most GNN frameworks by relying on graph convolutions. GAT introduces attention mechanisms that dynamically enable nodes in the graph to attend to their neighboring nodes during information aggregation. This attention mechanism

empowers GAT to assign different weights to nodes within a neighborhood, capturing more meaningful representations. By stacking multiple attention layers, GAT can effectively learn representations of nodes and capture complex patterns and dependencies within the graph data.

2.3 Rationale Learning

2.3.1 Normal Rationale Learning

Rationale learning is an approach aimed at improving the interpretability and reliability of machine learning models. One notable method in this field is LIME [58] by Ribeiro et al. (2016), which is a model-agnostic technique for generating local explanations for individual predictions. LIME produces sparse and interpretable explanations that highlight the most important features contributing to a prediction, facilitating a better understanding of the model's reasoning. Another method, Anchors [41] by Ribeiro et al. (2018), focuses on generating global explanations that identify the conditions under which a model's predictions change. Anchors produces simple and interpretable explanations that capture the most significant features and their interactions, surpassing other global explanation methods in performance. Additionally, Rationale Networks [29] by Lei et al. (2016) introduces a framework that jointly predicts the target variable and generates a binary rationale mask identifying important features. This rationale mask can be used to provide human-readable justifications for the model's predictions, enhancing comprehensibility and trust in the model.

2.3.2 Graph Rationale Learning

In the field of GNN interpretation with rationale learning, several papers have made significant contributions. One notable paper is Discovering Invariant Rationale (DIR) [51] by Wu et al. (2022). This paper introduces the concept of building intrinsically interpretable GNNs, emphasizing the importance of addressing data biases and identifying causal rationales that remain invariant across different distributions. The proposed approach leads to a significant improvement in the generalization ability of GNNs on graph classification tasks. Another paper is GNNInterpreter [50] by Wang et al. (2023). This paper presents a model-agnostic, model-level explanation method that utilizes a probabilistic generative graph distribution to explain the high-level decision-making process of GNNs. This method is well-known for its flexibility in generating explanation graphs with various node and edge features. It offers valuable insights into the interpretability of GNNs.

2.4 Sparse Regularization

As the complexity and scale of graph data continue to increase, some essential research directions have emerged, like sparse regularization and continual learning. Sparse regularization techniques encourage sparsity in the learned models, and continual learning helps GNNs to adapt incrementally to new tasks or evolving graph data. Elastic Net [35] by Xiaorui Liu et al. (2021) combines L1 and L2 regularization to leverage the benefits of both sparse regularization methods. By incorporating sparsity-inducing penalties and ridge-like regularization, Elastic Net balances feature selection and parameter shrinkage. This approach has been explored in GNNs to promote sparse representations while handling correlated features in graph data. Self-regularized graph neural network (SR-GNN) [14] by Zhan Gao et al.(2022) focuses on the stability of GNN and how they respond to perturbations in the graph data, which is important for guaranteeing reliable performance in noisy data. SR-GNN enhances the stability of the architecture by regularizing the filter frequency responses in the graph spectral domain. Combining spectral sparsification with Laplacian learning can also improve the scalability issue of graph-based learning [7].Continual Learning via Neural Pruning (CLNP) [16] by Siavash Golka et al.(2019) based on neuronal model sparsification. It trains new tasks utilizing the unused weights of the network, which takes advantage of the features learned by the previous tasks while generating zero catastrophic forgettings. Continual Learning with Node-Importance-based Adaptive Group Sparse Regularization (AGS-CL) [20] by Sangwon Jung et al.(2020) uses group sparsity-based penalties when learning each node of a neural network based on its importance, which is adaptively updated after learning each task.UD-GNN [36] by Yang Liu et al.(2022) also uses the Sparse regularization way to drop the weight parameters in the model, which is close to zero, and retrain to improve the model's performance.

2.5 Label Conformity In Graph Encoders

Homogeneous graphs in GNN are more like all nodes, and edges in the graph share the same feature space. Homogeneous graphs in GNN are more like all nodes, and edges in the graph share the same feature space. The key idea behind homogeneous GNNs is to aggregate information from neighboring nodes and update node representations iteratively. Like the GCN, GAT also belongs to homogeneous GNNs.Heterogeneous GNNs in graph datasets are designed to handle graphs with different types of nodes or edges, each with its own attributes, like the neighborhood node feature is different from the target node feature. HAN [49] by Xiao Wang et al.(2019) introduced a two-step attention mechanism to capture and aggregate information from different types of nodes and edges in a heterogeneous graph.HetGNN [59] by Chuxu Zhang et al.(2019) proposed a framework to address both structural and feature heterogeneity in a graph by employing type-specific aggregation and update functions. HGT [19] by Ziniu Hu et al.(2020) extended the Transformer architecture to address heterogeneous graphs. HGT

introduced a hierarchical aggregation mechanism to capture the structural information and semantic relationships in the graph. HetSANN [18] by Huiting Hong et al. (2019) applies a graph neural network with an attention mechanism to aggregate multi-relational information from the projected neighborhood. This enables the model to capture the diverse relationships and dependencies present in heterogeneous information networks. DisGNN [61] by Shaohua Fan et al.(2022) incorporates self-supervision tasks like capturing the heterogeneous edge features to guide automatic edge disentanglement. This framework encourages the model to capture the graph's distinguishable relations and neighborhood interactions.

Table 2.1: Mathematical Notations.

Symbol	**Description**
G	Input graphs
V	Node set
E	Edge set
GRv	Graph encoder layer module
H	Node representation matrix
A	Adjacency matrix
X	Node feature matrix
M_e	Edge attention matrix
M_n	Node attention matrix
C	Causal feature
S	Non-causal feature
C_R	Causal node representation
S_R	Non-causal node representation
C_{sub}	Causal subgraph
S_{sub}	Non-causal subgraph
C_{sub}^H	Causal high confidence subgraph
S_{sub}^H	Non-causal high confidence subgraph
C_{sub}^L	Causal low confidence subgraph
S_{sub}^L	Non-causal low confidence subgraph
L	Hop number
Y	Graph label
f_{pool}	Graph pooling function
f_G	Causal attention graph and no-causal attention graph generator
f_C	Causal classifier

Chapter 3

Preliminary

This chapter presents a formalization of the terminology used in the study, aiming to establish a clear foundation for the research. It includes the formalization of Graph Encoders (Section 3.1), Debiasing Graph Encoders (Section 3.2), and Graph Rationale Learning (Section 3.3). Additionally, in Section 3.4, we provide a formalization of the aspect-focused headline generation problem.

3.1 Graph Encoders

3.1.1 Graph representation learning

GNNs are trained by propagating information through the edges of a graph, with each node representing an entity and each edge representing a relationship. The features of each node are updated by aggregating information from neighboring nodes using the message-passing mechanism [15]. Several recent variants of GNNs have emerged, each integrates unique techniques to learn graph representations. Examples include GCN [22], GIN [53], and GAT [48].

3.1.2 Pooling Layers

GNNs rely on vectorial graph representation and require a crucial pooling layer for effective graph classification. The pooling layer is a parameterized function that maps multiple graph representation vectors to a single vector. Additionally, pooling operations can reduce the parameter size by down-sampling the nodes, generating smaller representations, and thus addressing issues like overfitting, permutation invariance, and computational complexity. Various variants of the graph pooling layer exist, including mean pooling, sum pooling, and cluster-based pooling [27, 4].

3.1.3 Attention Mechanism In GNNs

The attention mechanism in GNNs can be based on either nodes or edges in the graph data. For edge-level attention [21, 48], the GNN encoder weighs the edges and aggregates neighborhood information based on these edge weights to update the node representations. For node-level attention [24], the GNN encoder learns a self-attention mask that recognizes the nodes with the highest representation scores from the neighborhood, while masking the low-representation nodes during the aggregation process.

In the following equation, $A \in \mathbb{R}^{|V|\times|V|}$ represents the adjacency matrix, where $A[i,j] = 1$ if $(v_i, v_j) \in E$. H' denotes the node representations before the update. Equation 3.1 represents the edge-level attention, where $M_e \in \mathbb{R}^{|V|\times|V|}$ is the attention matrix that weights the edges in the graph. Equation 3.2 represents the node-level attention, where $M_n \in \mathbb{R}^{|V|\times 1}$ is the node attention matrix.

$$H = GRv(A \odot M_e, H') \tag{3.1}$$

$$H = GRv(A, M_n \odot H') \tag{3.2}$$

After getting all node representations, we can do the graph pooling. Here the pooling method we choose is sum pooling:

$$f_{pool}(H) = \sum_{h_i \in H[:,i]}^{|V|} h_i \tag{3.3}$$

3.2 Debasing Graph Encoders

Due to the message-passing attribute of GNNs, it is easier for them to aggregate biased information from neighboring nodes. This can lead to the aggregation of spurious correlations between non-causal features and the target variable. For example, in colored MNIST datasets, where the digit part is causal for labels, but the background color is highly correlated with the digit, such as 90% of the background of "zero" being red. GNNs can easily aggregate the background color information, which is non-causal, and use it to make predictions. However, the learning target of GNNs is to recognize the digits, not the background color. Therefore, debiasing graph encoders are needed to address these issues.

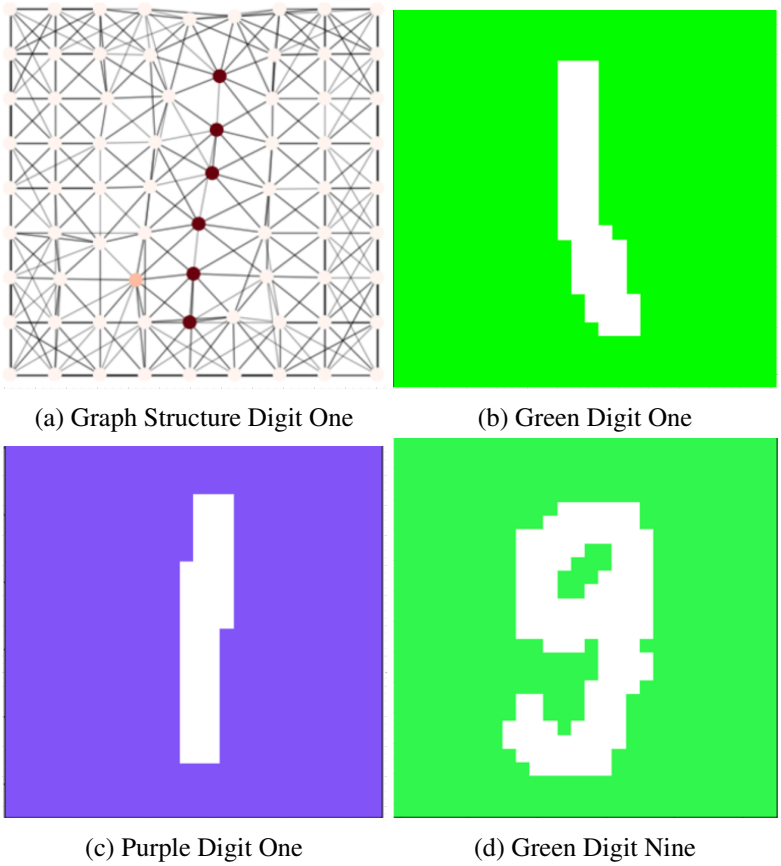
Debiasing graph encoders serve as a regularization technique for GNNs, with the objective of mitigating the influence of noisy or irrelevant features present in the graph. This regularization approach contributes to the generation of more robust and informative representations. The introduction of debiasing graph encoders is a recent proposal aimed at enhancing the performance and generalization capabilities of GNNs. Several frameworks have been developed to help GNNs mitigate the impact of bias. CAL (Causal Attention Learning) [46] by Yongduo Sui et al. (2022) employs attention modules to estimate causal and non-causal features in the

input graph, enhancing the interpretability and generalization capabilities of GNN classifiers. On the other hand, DisC (Disentangled Causal substructure) [43] by Shaohua Fan et al.(2022) utilizes a parameterized edge mask generator to explicitly separate the input graph into causal and biased subgraphs. Separate encoders are then trained to encode the representations of these subgraphs. DisGNN [61] by Tianxiang Zhao et al.(2022) automatically disentangles edges using three heuristics and pretext tasks. The edge disentanglement module in DisGNN captures distinguishable relations and neighborhood interactions, aggregating their outputs for node representations to mitigate the influence of biased data. GraphDE (Generative Framework for Debiased Learning and Out-of-Distribution Detection on Graphs) [33] by Zenan Li et al.(2022) proposes a unified probabilistic model that mathematically characterizes the distribution shifts in graph data by introducing a latent environment variable as an indicator and modeling the graph generative process accordingly. GraphDE successfully down-weights non-causal data during training.

3.3 Graph Rationale Learning

Graph Rationale Learning aims to offer an interpretable and easily comprehensible approach to demonstrate that GNN predictions rely on key features or a small subset of nodes in the graph, which contribute significantly to correct predictions [8, 24, 34]. Hence, the process of identifying the rationale part in graph data is referred to as Graph Neural Network Rationalization. However, many GNN models tend to rely on data biases when making predictions, particularly by utilizing shortcut features (non-causal part). These biases are susceptible to change outside the training distribution. Since GNNs possess the ability to make predictions based on subgraphs, we can leverage this property and integrate it with graph rationale learning to draw the following conclusion:

- The minimal form of the causal subgraph is capable of accurately predicting the ground-truth label of the graph. For instance, in Figure 3.1(a), which represents an image of the graph type "one", it is clear to identify the nodes that form the causal subgraph, and their combination constitutes the minimal form of the causal subgraph. Ideally, GNN models should be able to utilize this subgraph to make correct decisions. $i.e., P(Y|(min(|C_{sub}|), C_{sub})$.
- By using causal subgraphs to make predictions ($C_{sub} \rightarrow Y$), the relationship should remain invariant across different non-causal components (S_{sub}), meaning that $P(Y|C_{sub}, S_{sub}) = P(Y|C_{sub})$. For example, in Figure 3.1(b), if 90% of the background color of the digit one is green in the training data, we ideally want GNN models to use the shape of the digit one to make predictions regardless of changes in the background color. For instance, when predicting Figure 3.1(c), we expect the GNN to correctly identify it as a one without considering the background color.



(a) Graph Structure Digit One (b) Green Digit One

(c) Purple Digit One (d) Green Digit Nine

Figure 3.1: Example illustrating the property of graph rationale learning.

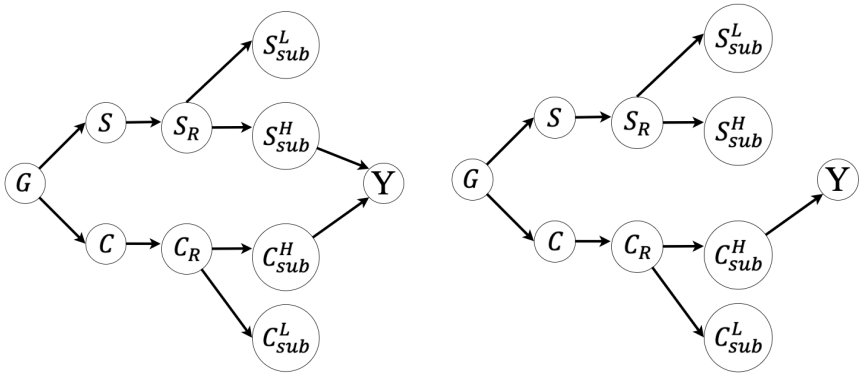


Figure 3.2: Structure Causal Model. (a): The structural causal model for graph classification. (b): The structural causal model for graph classification with our GNN debias framework.

- The non-causal component S_{sub} should be irrelevant for the prediction, meaning that $P(S_{sub}) = P(S_{sub}|C_{sub})$. For instance, in Figure 3.1(d), the background color of the digit nine is green in the testing data. However, in the training data, 90% of the background color of the digit one is green, as shown in Figure 3.1(b). If the model is biased, it may incorrectly predict Figure 3.1(d) as a one instead of a nine. This outcome is not desired and goes against our intentions.

3.4 Problem Formulation

To address the debiasing issue in GNNs, we adopt a causal perspective in our modeling approach. Figure 3.2 depicts the Structural Causal Model (SCM) [39] of our proposed framework, emphasizing the causal relationships among four variables: the input graph data G, the causal feature C, and the non-causal feature (shortcut feature) S. Given C and S, we derive the node representations for the causal feature C_R and the non-causal feature S_R. Subsequently, we obtain the high-confidence subgraphs (C^H_{sub}, S^H_{sub}) and the low-confidence subgraphs (C^L_{sub}, S^L_{sub}) from C_R and S_R, respectively. Finally, the model predicts the label Y. The connections between variables in Figure 3.2(a) represent cause-and-effect relationships, with the arrow indicating the direction of causality.

Before delving into the Structural Causal Model (SCM), we introduce four variables related to the confidence estimator. Firstly, the high-confidence non-causal subgraphs S^H_{sub} represent subgraphs derived from the input graph G that predominantly contribute to biased information. Conversely, the high-confidence causal subgraphs C^H_{sub} are obtained from the input graph G

and are crucial for making predictions. On the other hand, the low-confidence causal subgraphs C_{sub}^{L} are captured during training but have minimal impact on label prediction. Lastly, the low-confidence non-causal subgraphs S_{sub}^{L} are captured during training but do not transmit significant biased information. Overall, incorporating high-confidence and low-confidence subgraphs helps identify uncertain factors. Specifically, we provide the following explanations of the SCM:

- $\mathbf{C} \leftarrow \mathbf{G} \rightarrow \mathbf{S}$: The causal feature C accurately represents the intrinsic true feature of the graph data G. On the other hand, the non-causal feature S is susceptible to data bias and can vary easily in testing data. This explanation can be supported if both C and S coexist in the graph data G.

- $\boldsymbol{S_{sub}^{H} \leftarrow S_R \rightarrow S_{sub}^{L}}$: The model captures the non-causal representation S_R from the non-causal feature S. The high-confidence non-causal subgraphs S_{sub}^{H} are derived from the input graph G based on the non-causal representation S_R, and they predominantly contribute to transmitting biased information. On the other hand, the low-confidence non-causal subgraphs S_{sub}^{L} are also obtained from the non-causal representation S_R, but they do not significantly transmit biased information. The coexistence of S_{sub}^{H} and S_{sub}^{L} in G allows for learning from S_R, thereby representing true causal effects. In other words, the process of the GNN capturing features is uncertain.

- $\boldsymbol{C_{sub}^{H} \leftarrow C_R \rightarrow C_{sub}^{L}}$: The model captures the causal representation C_R from the causal feature C. The high-confidence causal subgraphs C_{sub}^{H} are derived from the input graph G based on the causal representation C_R and are crucial for making predictions. On the other hand, the low-confidence causal subgraphs C_{sub}^{L} are also obtained from the causal representation C_R, but they have minimal impact on label prediction. The coexistence of C_{sub}^{H} and C_{sub}^{L} in G allows for learning from C_R, thereby representing true causal effects.

- $\boldsymbol{C_{sub}^{H} \rightarrow \mathbf{Y} \leftarrow S_{sub}^{H}}$:In the presence of both C_{sub}^{H} and S_{sub}^{H} in the graph data G, the high-confidence non-causal variable S_{sub}^{H} can influence the high-confidence causal variable C_{sub}^{H} in predicting the label Y.

Based on the Structural Causal Model (SCM) depicted in Figure 3.2(a), traditional GNNs typically use both the non-causal variable S and the causal variable C for making predictions. The non-causal variable, as shown in Figure 1.1, is relatively easier to learn compared to the causal variable representing the shape of the digits. By leveraging this simple information, GNNs can quickly achieve low loss. However, in the SCM shown in Figure 3.2(a), we can observe that the child variable S_{sub}^{H} opens a backdoor path [40], represented as $\boldsymbol{C_{sub}^{H} \leftarrow C_R \leftarrow \mathbf{C} \leftarrow \mathbf{G} \rightarrow \mathbf{S} \rightarrow S_R \rightarrow S_{sub}^{H} \rightarrow \mathbf{Y}}$. Although C_{sub}^{H} does not have a direct connection to Y, there exists the non-causal variable S_{sub}^{H} between them, leading to a spurious correlation between the predictions of G and Y.

Therefore, it is essential to find a solution that breaks the backdoor path and enables the GNN to make predictions solely based on C_{sub}^H. By leveraging causal theory [39] and integrating it with graph rationale learning, we can address this issue. To effectively sever the backdoor path, we intervene on C_{sub}^H, which is further explained in detail in Chapter 3.3 of graph rationale learning. Let $do(C_{sub}^H)$ denote the intervention on C_{sub}^H.

- Expect non-causal part S_{sub}^H should be unnecessary for prediction. So S_{sub}^H and C_{sub}^H can be independent to each other: $P(S_{sub}^H|do(C_{sub}^H)) = P(S_{sub}^H)$
- Expect causal subgraph to make prediction : $C_{sub}^H \rightarrow Y$ should be invariant across different non-causal subgraph. So the non-causal subgraph S_{sub}^H should do nothing with causal subgraph : $P(Y|do(C_{sub}^H), S_{sub}^H) = P(Y|C_{sub}^H)$
- Expect causal subgraph in its minimal form can sufficiently predict the ground-truth graph label, because in this way can eliminate non-causal effects as much as possible: $P(Y|do(C_{sub}^H)) = P(Y|min(|C_{sub}^H|), C_{sub}^H)$

For a better understanding, we construct the other SCM for the target of our proposed GNN debias framework in Figure 3.2(b).

- $\mathbf{C} \leftarrow \mathbf{G} \rightarrow \mathbf{S}$. Same as Figure 3.2(a), the non-causal variable S and causal variable C coexist in graph data G.
- $\mathbf{G} \rightarrow \mathbf{C} \rightarrow \boldsymbol{C_R} \rightarrow \boldsymbol{C_{sub}^H} \rightarrow \mathbf{Y}$. The input graph data G is only based on the causal variable C to get Y.

Based on the above formulation, when we have the graph data G, our objective is to learn the causal subgraph $f_G : G \rightarrow C$ and the non-causal subgraph $f_G : G \rightarrow S$ simultaneously. We then utilize the causal subgraph to make accurate predictions for the graph label: $f_C : C \rightarrow Y$.

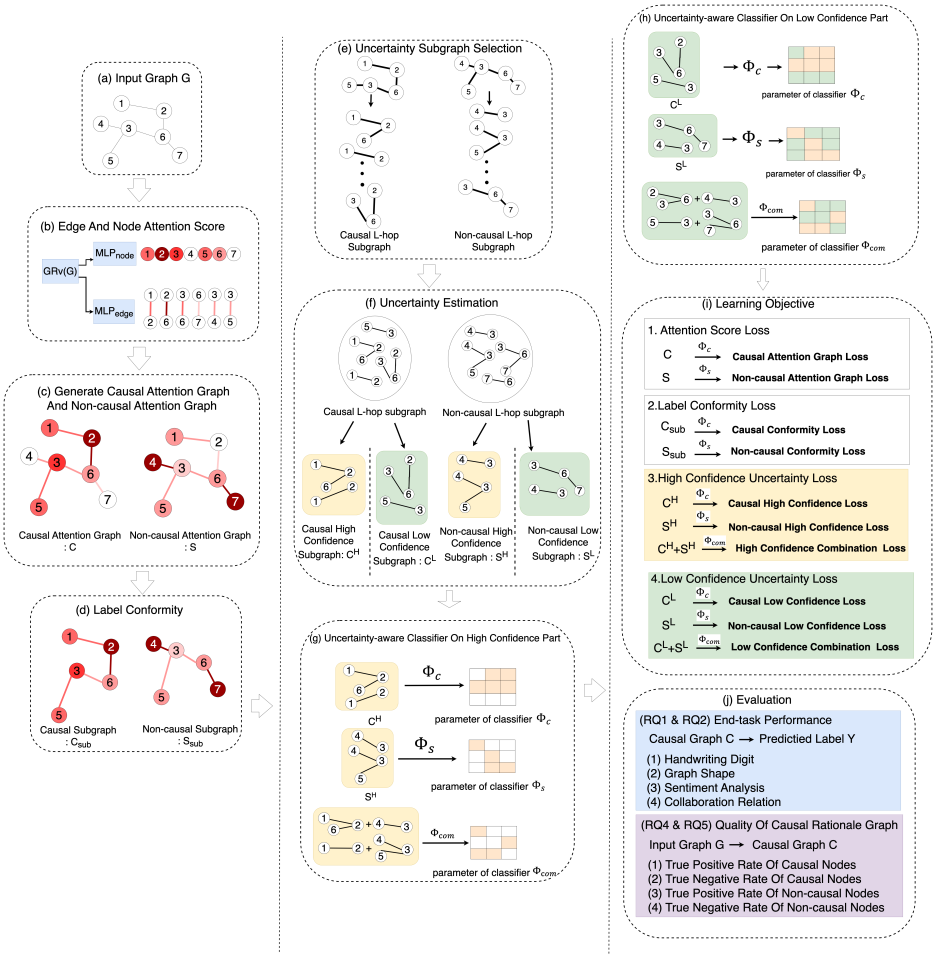


Figure 3.3: The architecture of the UACG framework. The red color denotes the attention score, where a deeper red indicates a higher attention score, and vice versa. The yellow color denotes the high confidence of causal subgraphs, non-causal subgraphs, causal classifier parameters, non-causal classifier parameters, and certainty learning objective. The green color denotes the low confidence of causal subgraphs, non-causal subgraphs, causal classifier parameters, non-causal classifier parameters, and certainty learning objective. The blue color denotes the target of RQ1 and RQ2 evaluation. The purple color denotes the target of RQ4 and RQ5 evaluation. The overall framework consists of the following steps. Step (b) Obtain the node attention score and edge attention score based on the input graph G. Step (c) constructs the causal attention graph and non-causal attention graph. Step (d) reconstructs the causal and non-causal subgraphs based on the label conformity perspective. Step(e) generate L-hop subgraphs based on each node in the causal and non-causal subgraphs obtained from step (d). Step (f) categorizes all L-hop subgraphs into high and low confidence. Step (g) is the training process of the high confidence certainty part, where parameters marked with white color are considered insignificant and masked with zero after the training process. Step (h) is the training process of the low confidence certainty part, which only adjusts the number of insignificant parameters from step (g). Finally, steps (i) and (j) encompass the overall learning objectives and evaluation plan.

Chapter 4

Methodology

Motivated by the analysis presented in Section 3.4, this Chapter introduces the UACG framework, which aims to debias GNNs and alleviate the impact of non-causal graphical content on graph classification tasks. The overall framework is depicted in Figure 3.3. We elaborate on each component of the proposed UACG framework and subsequently describe the components of our learning objectives.

4.1 Overview of Uncertainty-aware Causal Graph Learning

Figure 3.3 presents a summary of the overall UACG framework, which can be broken down as follows. First, we compute the node and edge attention scores based on the input graph G and construct the causal and non-causal attention graphs. These graphs represent the importance of nodes and edges in capturing causal and non-causal information, respectively. Second, we employ label conformity techniques to disentangle the different characteristics of nodes that influence each other. This process involves reconstructing the causal and non-causal attention graphs into subgraphs that specifically capture causal and non-causal relationships. Third, we generate L-hop subgraphs based on each node in the causal and non-causal subgraphs. These subgraphs are used for uncertainty estimation, where we perform subgraph selection based on uncertainty scores. In Figure 3.3(e), we illustrate this process for a hop number of one, for ease of understanding. After obtaining the sets of causal and non-causal L-hop subgraphs, we split each subgraph into high and low uncertainty categories using uncertainty estimation. Once the training process for the high uncertainty subgraphs is complete, the attention scores for the causal and non-causal parts will significantly differ, effectively removing the correlation between them in the graph. After completing the training of the high confidence uncertainty part, we ensure the model's completeness by applying sparse regularization techniques to mask insignificant parameters in the classifiers and retrain the model to compensate for the low uncertainty. In this work, the generation of the causal attention graph, label conformity, intervention, and uncertainty

estimation primarily contribute to the development of the high-confidence uncertainty-aware classifier. As a result, the framework can be divided into two training parts: the high confidence uncertainty part ($\mathcal{L}^{H}_{all-uncertainty}$) and the low confidence uncertainty part ($\mathcal{L}^{L}_{all-uncertainty}$). Thus far, we have achieved our overall objective as shown in Equation 4.1. The details of the UACG framework will be further discussed in the following section.

$$\min \mathcal{L}^{H}_{all-uncertainty} + \mathcal{L}^{L}_{all-uncertainty} \tag{4.1}$$

4.2 Causal Graph Generation

To effectively identify a rational causal subgraph for accurate predictions, it is crucial to separate the causal and non-causal features from the original input graph G. This process corresponds to Figure 3.3 (a) to Figure 3.3 (c). To facilitate a more intuitive and convenient understanding of Section 4.2, a flowchart summarizing the content is presented in Figure 4.1 of Section 4.2.

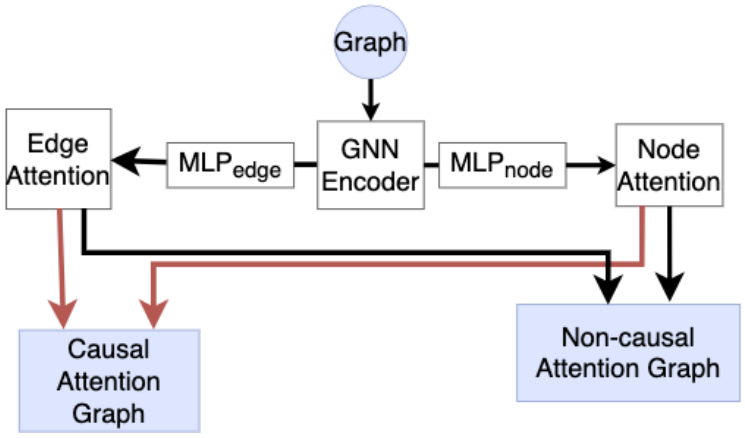


Figure 4.1: Causal Graph Generation Flow Chart.

4.2.1 Causal and Non-causal Attention Graph

In regular attention-based GNN models [26, 47, 28, 32], the estimation of node representations focus solely on either the node level or the edge level. However, in UACG, inspired by CAL [46], both node-level and edge-level attention scores are incorporated.

Given a graph G with the adjacency matrix A and node feature X, denoted as $G = A, X$, we input this graph G into the GNN encoder GRv. The node representations H are then produced using Equation 4.2.

$$H = GRv(G) \tag{4.2}$$

For a node $v_i \in V$ and an edge $(v_i, v_j) \in E$, where h_i denotes the representation of node v_i. UACG generate the node and edge attention by using MLP_{node} and MLP_{edge}:

$$\alpha_{ci}, \alpha_{si} = \sigma(MLP_{node}(h_i)) \tag{4.3}$$

$$\beta_{cij}, \beta_{sij} = \sigma(MLP_{edge}(h_i || h_j)) \tag{4.4}$$

The $||$ represents the concatenation operation. α_{ci} and α_{si} represent the node attention scores for the causal and non-causal parts, respectively. β_{cij} and β_{sij} represent the edge attention scores for the causal and non-causal parts, respectively. In order to make the causal and non-causal attention graph easier to distinguish, based on equation 4.3 and equation 4.4, UACG uses the σ, which represents the softmax function to introduce the complementary concept to make the $\alpha_{ci} + \alpha_{si} = 1$ and $\beta_{cij} + \beta_{sij} = 1$. UACG use all attention score to formulate causal node attention matrix: $M_n = \{\beta_{ci} | i \in V\}$, non-causal node attention matrix: $\overline{M_n} = \{\beta_{si} | i \in V\}$, causal edge attention matrix: $M_e = \{\beta_{cij} | (i,j) \in E\}$, and non-causal edge attention matrix: $\overline{M_e} = \{\beta_{sij} | (i,j) \in E\}$. After preparing all matrices, we constructed the causal attention graph in equation 4.15 and the non-causal attention graph in equation 4.16.

$$C = \{A \odot M_e, X \odot M_n\} \tag{4.5}$$

$$S = \{A \odot \overline{M_e}, X \odot \overline{M_n}\} \tag{4.6}$$

4.2.2 Causal and Non-causal Attention Graph Estimation

We wish the causal C and non-causal S attention graphs better capture the node information and become more different during the training. So here we use two GNN encoder layers, one for the causal attention graph and one for the non-causal attention graph, to get the representations and make predictions based on graph pooling and classifiers:

$$\hat{y}_c = \Phi_c(f_{pool}(GRv_c(C))) \tag{4.7}$$

$$\hat{y}_s = \Phi_s(f_{pool}(GRv_s(S))) \tag{4.8}$$

However, the objectives of the causal and non-causal attention graphs are different. The objective of the causal attention graph is to approach the ground truth Y, while the objective of the non-causal attention graph is to avoid contributing any valuable information that could perturb the causal part. Inspired by CAL [46], we define the loss functions for the causal attention graph $\mathcal{L}_C$ and the non-causal attention graph $\mathcal{L}_S$ as follows:

$$\min_{\alpha,\beta,\Phi_c} \mathcal{L}_C = -\frac{1}{|D|} \sum_{G \in D} y_G \log(\hat{y}_c) \tag{4.9}$$

$$\min_{\alpha,\beta,\Phi_s} \mathcal{L}_S = -\frac{1}{|D|} \sum_{G \in D} KL(Unif, \hat{y}_s) \tag{4.10}$$

The equation 4.9 represents the standard cross-entropy loss based on the dataset D. In equation 4.10, the objective is to make the non-causal attention graph tend to classify into any label. To achieve this, we drive its predictions to be evenly distributed across all categories using a uniform distribution. The term KL represents the KL-Divergence, which is used to make the non-causal prediction distribution closer to the uniform distribution. In order to enhance the model's robustness to the data, we have incorporated ideas from CAL, which is a state-of-the-art GNN debiasing framework. We have adopted CAL's approach of obtaining attention scores for both nodes and edges together, as well as the idea of approaching a uniform distribution. However, we have omitted one additional function from CAL, which involves randomly combining the causal and non-causal representations based on the same data batch and making predictions. We have excluded this step because, during the training process with biased data, the GNN model is initially biased. If the learning of the causal part representation is biased, the combination of predictions will also be biased.

4.3 Label Conformity Estimation

This section corresponds to Figure 3.3(d). GNNs typically use the message-passing attribute to update node representations based on their neighborhoods. However, the neighborhoods of nodes can sometimes contain different features, leading GNNs to potentially incorporate biased information. For example, in Figure 3.1(a), ideally, we want the combination of nodes with the same shape to make a prediction, but GNNs may inadvertently aggregate background nodes as well. To address this issue, UACG introduces label conformity to prevent GNNs from aggregating biased information at an early stage. Label conformity estimation is a method used to identify homogeneous and heterogeneous edges in the graph. Further details will be discussed below.

4.3.1 Homo-edges and Hetero-edges Definition

If the features carried by certain nodes are very similar, we can consider the edges between these nodes as homophilous (homo-edges). On the other hand, if the features are different, we refer to the edges as heterophilous (hetero-edges). Below are the pseudo edge sets that distinguish between homo-edges and hetero-edges:

$$E^{homo} = \{A_{i,j} | A_{i,j} \neq 0, X_i \sim X_j\} \tag{4.11}$$

$$E^{hetero} = \{A_{i,j} | A_{i,j} \neq 0, X_i \not\sim X_j\} \tag{4.12}$$

The notation $X_i \sim X_j$ indicates that the feature X_i is similar to the feature X_j.

4.3.2 Homo-edges and Hetero-edges Recognition

Here, I propose using label conformity to improve the model's generalization and accurately identify the rational causal subgraphs by learning the connection information and masking all the hetero-edges in the graph. To estimate the node connection information, I employ an MLP (MLP_{hetero}) with the following equation:

$$e_{ij}^{hetero} = sigmoid(MLP_{hetero}(h_i||h_j)) \tag{4.13}$$

In equation 4.13, the hetero-weight is assigned to each edge. The value of e^{hetero} is in the range of (0,1). If the hetero-weight of any edge is greater than 0.5, UACG considers this connection to be heterophilous and creates a mask to mask the hetero-edges:

$$M_{hetero}[i,j]_{(i,j)\in E} = \begin{cases} 0, & \text{if } e_{ij}^{hetero} > 0.5 \\ 1, & \text{otherwise} \end{cases} \tag{4.14}$$

Use such masker to reconstruct the causal subgraph and non-causal subgraph below:

$$C_{sub} = \{A \odot M_{hetero} \odot M_e, X \odot M_n\} \tag{4.15}$$

$$S_{sub} = \{A \odot M_{hetero} \odot \overline{M_e}, X \odot \overline{M_n}\} \tag{4.16}$$

4.3.3 Hetero-edges Estimation

In Figure 4.2(b), in the ideal scenario, all hetero-edges can be identified as shown in Figure 4.2(a), allowing for the correct separation of the rational causal graph. Unfortunately, directly identifying the hetero-edges is not straightforward since we do not have ground truth labels for them. However, our objective is to identify the rational causal nodes. We can assume that if the predictions of the masked causal subgraph and masked non-causal subgraph approach the ground truth labels during training, the identification of hetero-edges is also improving. With that in mind, we can formulate the loss function as follows:

$$\hat{y}_{CE} = \Phi_c(f_{pool}(GRv_c(C_{sub}))) \tag{4.17}$$

$$\hat{y}_{SE} = \Phi_s(f_{pool}(GRv_s(S_{sub}))) \tag{4.18}$$

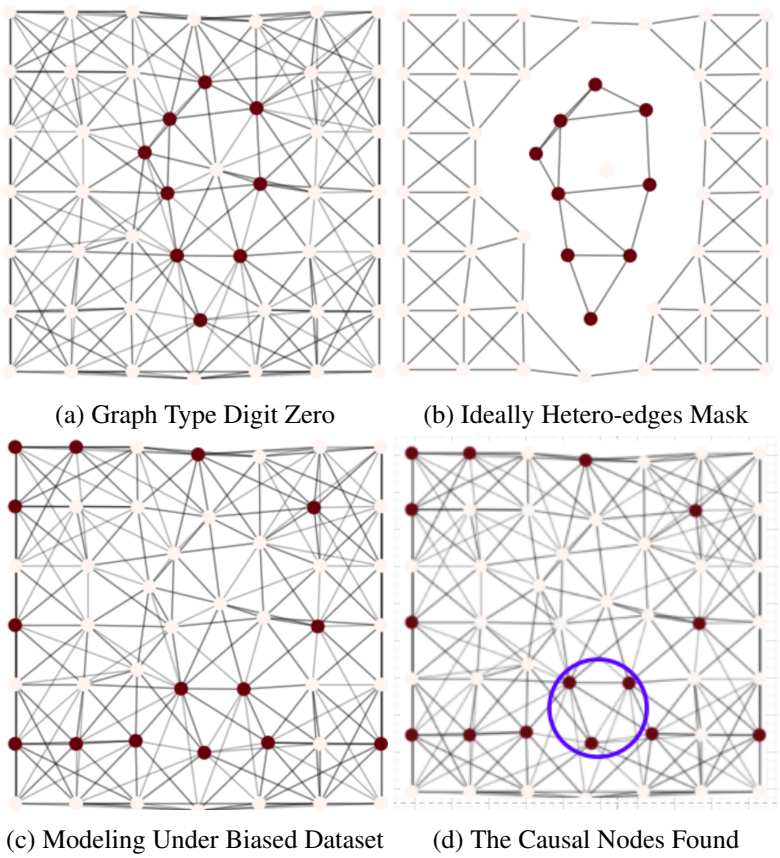
$$\min_{e^{hetero}} \mathcal{L}_{CE} = -\frac{1}{|D|}\sum_{G\in D} y_G \log(\hat{y}_{CE}) \tag{4.19}$$

$$\min_{e^{hetero}} \mathcal{L}_{SE} = -\frac{1}{|D|}\sum_{G\in D} KL(Unif, \hat{y}_{SE}) \tag{4.20}$$

And the overall label conformity loss is :

$$\mathcal{L}_{conform} = \mathcal{L}_{SE} + \mathcal{L}_{CE} \tag{4.21}$$

This method does not only help to find the hetero-edges but also can help to disentangle the causal and non-causal features.



(a) Graph Type Digit Zero (b) Ideally Hetero-edges Mask

(c) Modeling Under Biased Dataset (d) The Causal Nodes Found

Figure 4.2: Zero Graph Data.

4.4 Uncertainty Subgraph Sampling

Due to the presence of uncertainty during the training process with biased data, achieving a perfect result like Figure 4.2(b) is not feasible. Figure 4.2(c) represents the actual output of the causal part during training with biased data. To address this uncertainty, we introduce uncertainty estimation techniques after the intervention stage, as illustrated in Figure 3.3(e).

From Figure 4.2(c), it can be observed that the causal subgraph captured by the GNN appears to be randomly distributed throughout the graph. Although most of the nodes captured by the GNN are located in the background color nodes, we can still see that the GNN is attempting to identify the causal part. Based on this observation, subgraph sampling is a promising approach to prepare for uncertainty estimation.

After conducting several experiments and drawing inspiration from [13], we have decided to sample subgraphs using the L-hop subgraph concept. The L-hop technique utilizes breadth-first

search (BFS) to explore nodes within L layers of the source node. This process results in the discovery of an ego-net subgraph centered around the source node. Figure 3.3(e) provides an example of 1-hop subgraphs.

By using label conformity, we have already constrained the causal and non-causal parts to specific ranges. Under this condition, the ego-net of the source node only selects nodes with similar features to the source node, as the features within the causal part tend to be more similar than those in the non-causal part. By capturing the ground truth causal nodes, the GNN can also identify additional causal nodes using the L-hop technique. This approach provides an opportunity to capture nodes belonging to the ground truth causal part that may not have been initially captured by the biased GNN model during the early stages of training.

Ideally, we would like to obtain L-hop subgraphs based on each causal and non-causal subgraph node. However, this would be highly inefficient, as it would require performing as many classifications as there are nodes in the graph to evaluate the uncertainty. To address this, we introduce two methods for sampling causal part nodes in the uncertainty subgraph sampling section.

The first method is to select the Top-K nodes with the most significant node attention scores and construct k L-hop subgraphs. The second method, inspired by an algorithm [23], utilizes Locality Sensitive Hashing (LSH), which is a well-known algorithm used in approximate nearest neighbor search. In this case, we input the final node representations of each node in the causal subgraph into the LSH algorithm, which groups similar representations together into the same bucket. We then randomly select one node from each bucket and construct L-hop subgraphs based on these nodes. This process can be likened to finding the best apple tree. We first classify each apple based on its appearance and group apples with similar appearances together. Since the apples in each group are very similar, we randomly select one apple from each group. Based on the selected apples, we identify their corresponding apple trees to determine the ideal apple tree. This method saves time and introduces randomness, providing a better chance of overcoming the biases present in the data. More details and comparisons of these two methods can be found in Section 5.6. Non-causal node sampling, on the other hand, is simply performed by random sampling.

4.5 Uncertainty Estimation

This stage corresponds to Figure 3.3(f). After the uncertainty subgraph selection stage, we obtain a set of causal and non-causal L-hop subgraphs. Before proceeding to the Uncertainty-aware classification stage, we need to divide these L-hop subgraphs into high-confidence and low-confidence parts. For simplicity, we calculate the loss of each L-hop ego network by inputting

them into our graph classifier:

$$\mathcal{L}_{ego}^{C} = y_G \log(\Phi_c(C_{ego})) \tag{4.22}$$

$$\mathcal{L}_{ego}^{S} = KL(Unif, (\Phi_s(S_{ego}))) \tag{4.23}$$

C_{ego} represents each ego network of the causal part, while S_{ego} represents each ego network of the non-causal part. The loss value does not directly impact the model optimization but serves as a signal to indicate which ego network may lead us to the "ideal apple tree". However, things are not always straightforward, as discussed earlier, due to the GNN's tendency to rely on shortcuts (non-causal part) for predictions. If the value of equation 4.22 is too small, it may indicate that the captured pattern is a shortcut. On the other hand, if the value is significant, it suggests that the model may have captured incorrect information. To address this, we sort all the outputs of equation 4.22 and remove the 20% of ego networks with the smallest and largest loss values. Only the intermediate values are considered as high confidence, while the remaining ego networks are categorized as low confidence in the causal part. The selection of high and low confidence parts in the non-causal part is straightforward. If the value of equation 4.23 is small, the ego network associated with that value is classified as low confidence. Conversely, if the value of $\mathcal{L}_{ego}^{S}$ is closer to the uniform distribution, it means that the ego network does not exhibit a strong prediction trend towards any specific label, leading to its classification as low confidence. Referencing Figure 4.2(d), the nodes within the blue circle are more likely to lead us to the "ideal apple tree", and thus the ego networks associated with those nodes can be classified as high-confidence causal subgraphs.

4.6 Uncertainty-aware Parameter Learning

We have partitioned the graph into four segments: high-confidence causal subgraphs, low-confidence causal subgraphs, high-confidence non-causal subgraphs, and low-confidence non-causal subgraphs. This division enables us to specifically target and optimize the high-confidence and low-confidence segments. The first part corresponds to Figure 3.3(g). In this step, UACG feeds both the high-confidence causal and non-causal subgraphs into the classifier and calculates the high-confidence uncertainty loss:

$$\min_{\alpha,\beta,\Phi_c} \mathcal{L}_{uncer}^{HC} = -\frac{1}{|D|} \sum_{G \in D} y_G \log(\Phi_c(C_{ego}^{H})) \tag{4.24}$$

$$\min_{\Phi_s} \mathcal{L}_{uncer}^{HS} = -\frac{1}{|D|} \sum_{G \in D} KL(Unif, \Phi_s(S_{ego}^{H})) \tag{4.25}$$

Since we want the causal part to drive the optimization of the model, in equation 4.25, we block partial message propagation, ensuring that this value only influences the non-causal classifier. At this point, we have obtained the rationale subgraphs that are likely to contribute significantly.

To enhance the model's generalization and eliminate the backdoor path shown in Figure **??**, we further formulate the following equation:

$$G_{com}^{H} = random(c_{ego}^{H} + s_{ego}^{H}) \tag{4.26}$$

$$\min_{\alpha,\beta,\Phi_{com}} \mathcal{L}_{uncer}^{HCom} = -\frac{1}{|D|}\sum_{G\in D} y_G \log(\Phi_{com}(G_{com}^{H})) \tag{4.27}$$

$$\min \mathcal{L}_{uncer}^{H} = \mathcal{L}_{uncer}^{HC} + \mathcal{L}_{uncer}^{HS} + \mathcal{L}_{uncer}^{HCom} \tag{4.28}$$

In equation 4.26 , $c_{ego}^{H} \in C_{ego}^{H}$,$s_{ego}^{H} \in S_{ego}^{H}$. Inspired by sparse regularization [36, 20], we add capacity to the model for the low-confidence nodes while limiting the complexity of the high-confidence nodes of the biased model. This helps determine whether the uncertainty nodes contain helpful information for GNN predictions. As shown in Figure 3.2(a), the low-confidence nodes typically do not transmit any useful or biased information, but they coexist with other parts of the graph. Therefore, we prune the insignificant model parameters that were optimized based on the high-confidence uncertainty loss, and set these parts of the model parameters to zero. We then retrain the model on the low-confidence uncertainty loss, while keeping the significant model parameters from the high-confidence optimization frozen.

$$G_{com}^{L} = random(c_{ego}^{L} + s_{ego}^{L}) \tag{4.29}$$

$$\min_{\Phi_c} \mathcal{L}_{uncer}^{LC} = -\frac{1}{|D|}\sum_{G\in D} y_G \log(\Phi_c(C_{ego}^{L})) \tag{4.30}$$

$$\min_{\Phi_s} \mathcal{L}_{uncer}^{LS} = -\frac{1}{|D|}\sum_{G\in D} KL(Unif, \Phi_s(S_{ego}^{L})) \tag{4.31}$$

$$\min_{\Phi_{com}} \mathcal{L}_{uncer}^{LCom} = -\frac{1}{|D|}\sum_{G\in D} y_G \log(\Phi_{com}(G_{com}^{L})) \tag{4.32}$$

$$\tag{4.33}$$

Freezing the significant model parameters during the low-confidence training can effectively block the model becomes more uncertain.

4.7 Learning Objective

Finally, the learning objective of UAGC can be formulated as below two losses:

$$\mathcal{L}_{all}^{H} = \mathcal{L}_{C} + \mathcal{L}_{S} + \mathcal{L}_{conform} + \mathcal{L}_{uncer}^{H} \tag{4.34}$$

$$\mathcal{L}_{all}^{L} = \mathcal{L}_{uncer}^{LC} + \mathcal{L}_{uncer}^{LS} + \mathcal{L}_{uncer}^{LCom} \tag{4.35}$$

The learning objective for high-confidence causal graph in Equation 4.34 encourages Φ_c to accurately predict the causal part of the graphs, while encouraging Φ_s to uniformly predict the

non-causal part of the graphs. The parameters α and β help identify the rational parts in the graph, while e^{hetero} is used to recognize the hetero-edges.

The learning objective for low-confidence graph in Equation 4.35 aims to improve the plasticity of the high-confidence model and recover the model's capacity at uncertain nodes. Additionally, it helps to determine whether helpful information is hidden in the uncertain nodes.

Chapter 5

Experiments

This Chapter reports the experiments of our study in detail. We formulate five research questions to validate our proposed model. We present the four benchmark datasets used in our experiments in Section 5.2. To demonstrate the efficacy of our proposed method, we conducted a series of experiments on four benchmark datasets against two dominant types of models to quantitatively answer respective research questions in Sections 5.4, 5.5, 5.6 and 5.7. Lastly, we report our qualitative study in Section 5.8. The five research questions are formulated as follows:

5.1 Research Questions

- **RQ1**: How is the quantitative comparison of the effectiveness of generated causal graphs across synthetic benchmark datasets for the end-task of graph classification?
- **RQ2**: How is the quantitative comparison of the effectiveness of generated causal graphs across real-world benchmark datasets for the end-task of graph classification?
- **RQ3**: How does the key component contribute to the end-task of graph classification?
- **RQ4**: How is the quantitative comparison of the quality of generated causal graphs?
- **RQ5**: How is the qualitative comparison of the quality of generated causal graphs?

5.2 Datasets

We utilize two synthetic datasets and two real-world datasets for graph classification tasks. Here, we provide a brief introduction to the datasets, while the detailed dataset statistics are summarized in Table 5.1, Table 5.2, and Table 5.3.

Dataset	Causal Subgraph Type	Non-causal Subgraph Type
CMNIST-75sp [2]	Digit Subgraph	Color Background Subgraph
Spurious-Motif [55]	House/Cycle/Grid/Diamond Shape	BA/Tree Shape

Table 5.1: The content of the causal and non-causal subgraphs in the synthetic dataset.

Dataset	**Data Split**	**Bias degree**	**#Graphs**	**#Classes**	**#Avg. Nodes**	**#Avg. Edges**
CMNIST-75sp [2]	train	0.8/0.9/0.95	10K	10	65.7	505.7
	validation	0.5	5K	10	65.5	503.6
	test	0.1	10K	10	65.4	471.7
Spurious-Motif [55]	train	0.5/0.7/0.9	5,600	4	283.5	471.6
	validation	0.5/0.7/0.9	800	4	232.3	770.5
	test	0.5/0.7/0.9	1,600	4	283.5	770.5

Table 5.2: The detail of synthetic datasets.

Datasets	Domain	#Graphs	#Nodes	#Edges	#Classes
Graph-SST2 [57]	NLP	43,779	17.7	18.3	2
COLLAB [38]	Collaboration network	5,000	74.4	2467.7	3

Table 5.3: Real-world Dataset.

5.2.1 Spurious-Motif

There are currently multiple variants of the Spurious-Motif dataset. In this experiment, following the approach in [55], we created a synthetic dataset for graph classification. The dataset consists of 8,000 samples with four classes. Each graph in this dataset is composed of two parts: a non-causal subgraph and a causal subgraph. There are two types of non-causal subgraphs: BA-SHAPES and Tree. The BA-SHAPES subgraph is a Barabási-Albert (BA) graph [3], while the causal subgraph can be one of four types: House, Cycle, Grid, or Diamond. The visualizations of these subgraphs are shown in Figure 5.1. The task is to predict the type of the causal subgraph in the combined graph. In this experiment, we choose the Cycle class to define the bias using Equation 5.1:

$$b = \frac{\#Cycle_{Tree}}{\#Cycle} \tag{5.1}$$

Here, $\#Cycle_{Tree}$ represents the number of graphs with Cycle causal subgraphs combined with Tree non-causal subgraphs, and $\#Cycle$ denotes the total number of graphs in the Cycle class. Following [55], we set the proportion of Tree in the other three classes to $1 - b$.

5.2.2 CMNIST-75sp

Following [43], we constructed a biased MNIST image dataset similar to [2]. In this dataset, each category of digit image is highly correlated with the color of the digit image background. For instance, in the training set, 90% of the digit "one" images have a green background color, as shown in Figure 5.2, while the remaining 10% of digit "one" images have random background colors. In this case, the bias degree is 0.9. For the testing set, we constructed an unbiased test set where the digit labels are uncorrelated with the background colors. The goal of this test set is to evaluate whether the model can rely on the intrinsic digit signals for prediction.

5.2.3 Graph-SST2

Graph-SST2 is a graph-based version of the Stanford Sentiment Treebank dataset [57], which serves as a widely used benchmark for sentiment analysis tasks. In this dataset, each sentence is labeled with its corresponding sentiment, either positive or negative. The graph-based representation of the SST-2 dataset provides a more structured and interconnected view of the sentences, capturing the dependencies and relationships between the words.

5.2.4 COLLAB

The COLLAB dataset [38] focuses on the task of graph classification. It consists of graphs that represent scientific collaborations between researchers. Each graph represents a collaboration network, where the nodes represent researchers and the edges indicate collaborations between them. The objective of working with the COLLAB dataset, which has three classes, is to predict

the class or category of a researcher based on their collaboration network and potentially other features associated with the nodes or edges.

5.3 Experimental Settings

5.3.1 Baselines

The models we selected for comparison can be divided into two parts. The first part consists of three regular GNN models without any debiasing process. The second part includes two state-of-the-art GNN frameworks. Here is a detailed description of these models:

Without debias encoder:

- **GCN** [22]: GCN is a well-known variant of GNN. The fundamental idea behind GCN is to apply convolutional operations to graph data, analogous to how convolutional neural networks process images. GCNs utilize a neighborhood aggregation strategy to update node representations by combining information from neighboring nodes.
- **GAT** [48]: One of the most representative variants of GNN that utilizes attention mechanisms for representation learning. Attention mechanisms enable GAT to assign different weights to neighboring nodes, allowing it to focus on more relevant nodes in the vicinity. GAT computes a weighted sum of the features of neighbor nodes using an attention mechanism to determine the importance of each neighbor.
- **GIN** [53]: A variant of GNN based on graph isomorphism, which means that if two graphs have the same structure, they should have the same representation. GIN updates a node's representation by aggregating the features of its neighboring nodes and applying an MLP to combine the features. The output of the MLP is then added to the original node features to update the node's representation.

With debias encoder:

- **DIR** [51]: DIR is a GNN debias encoder that is inherently interpretable. It identifies causal rationales that remain invariant across multiple interventional distributions created through interventions on the training data. DIR focuses on stable causal patterns and filters out spurious patterns (i.e., non-causal patterns).
- **CAL** [46]: CAL is the best GNN framework that incorporates debias encoders. Its primary goal is to identify causal patterns and mitigate the confounding effect of shortcuts. CAL achieves this by estimating causal and shortcut features through attention modules, while also promoting stable relationships between causal estimation and prediction.

5.3.2 Metrics

We adopt **accuracy (ACC)** to measure the performance across different models in RQ1 and RQ2, as it corresponds to our expectation of using the causal graph learned by GNN to make accurate predictions. For RQ4 and RQ5, the evaluation focuses on the quality of the causal and non-causal subgraphs captured by different graph encoder variants. To assess the quality of the causal subgraphs, we utilize the **true positive rate (TPR)** and **positive predictive value (PPV)**. Similarly, the **true negative rate (TNR)** and **negative predictive value (NPV)** are used to evaluate the quality of the non-causal subgraphs. For evaluation purpose, we consider the causal part as a positive target class and the non-causal part with a negative class. The definitions of these metrics are formulated as follows: Note that the true positives as TP and true negatives as TN. Similarly, false positives are denoted as FP, and false negatives as FN.

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

$$TPR = \frac{TP}{TP + FN} \tag{5.3}$$

$$PPV = \frac{TP}{TP + FP} \tag{5.4}$$

$$TNR = \frac{TN}{TN + FP} \tag{5.5}$$

$$NPV = \frac{TN}{TN + FN} \tag{5.6}$$

5.3.3 Implementation Details

As for the training parameters, we train the models for 100 epochs with a batch size of 32 and use the Adam optimizer to optimize all models. In the proposed UAGC and CAL, we utilize GAT as the GNN encoders with three layers and 32 hidden units. For the intervention in Section 4.4, we select the Top-K significant causal and non-causal nodes, with a value of 20% for causal nodes and 80% for non-causal nodes. We set $L = 5$ in the generation of L-hop subgraphs and use 5 LSH buckets. We follow the default settings from the original papers and replicate the results for all the baselines. We report the average value and standard deviation of 5 runs for all the compared methods. All experiments are performed on Google Colab PRO.

5.4 Performance On Synthetic Datasets (RQ1)

To better observe the performance of GNN models on biased data, we conducted experiments using synthetic datasets and real-world datasets. Synthetic datasets allow us to more intuitively understand the changes in GNN models during training, while real-world datasets pose chal-

	CMINST-75sp			Spurious-Motif		
Model	b=0.8	b=0.9	b=0.95	b=0.5	b=0.7	b=0.9
GCN	$47.862_{\pm 1.66}$	$40.62_{\pm 2.37}$	$23.792_{\pm 0.28}$	$91.674_{\pm 1.27}$	$92.458_{\pm 0.38}$	$89.924_{\pm 0.54}$
GIN	$36.824_{\pm 1.43}$	$31.536_{\pm 0.97}$	$18.852_{\pm 1.74}$	$\mathbf{97.262_{\pm 0.2}}$	$\mathbf{98.134_{\pm 0.01}}$	$90.524_{\pm 5.66}$
GAT	$74.456_{\pm 6.61}$	$63.71_{\pm 4.83}$	$44.846_{\pm 2.03}$	$91.80_{\pm 0.06}$	$92.934_{\pm 0.08}$	$88.099_{\pm 0.23}$
DIR	$9.766_{\pm 0.03}$	$9.644_{\pm 0.05}$	$10.317_{\pm 0.02}$	$34.525_{\pm 6.06}$	$27.277_{\pm 1.02}$	$24.83_{\pm 0.04}$
CAL	$51.714_{\pm 5.4}$	$38.588_{\pm 2.75}$	$24.68_{\pm 8.09}$	$90.625_{\pm 7.64}$	$94.279_{\pm 1.03}$	$\mathbf{92.474_{\pm 0.47}}$
UACG	$\mathbf{74.912_{\pm 1.61}}$	$\mathbf{64.81_{\pm 3.96}}$	$\mathbf{47.381_{\pm 4.39}}$	$93.214_{\pm 1.06}$	$89.437_{\pm 7.14}$	$88.059_{\pm 3.09}$

Table 5.4: The ACC(%) of graph classification for synthetic datasets. The best results are highlighted in bold.

lenges in directly observing the non-causal data. However, understanding the debiasing process is crucial for guiding the GNN debias task.

Setup. In this section, we used two synthetic datasets: Spurious-Motif and CMINST-75sp. The Spurious-Motif dataset had data bias degrees of 0.5, 0.7, and 0.9, following the experimental settings in [46]. The CMINST-75sp dataset had data bias degrees of 0.8, 0.9, and 0.95, following the experimental settings in [43]. CMINST-75sp presented a more challenging scenario as the causal and non-causal parts of each handwritten digit in the dataset were constantly changing. The Spurious-Motif dataset, on the other hand, was relatively simpler with only two causal patterns and four non-causal patterns. The overall results are presented in Table 5.4.

Results. Analyzing Table 5.4, we observe that the performance gap is not significant for the Spurious-Motif dataset, except for DIR. The dataset's composition may not be complex, resulting in good performance for most GNN models without debiasing, particularly GIN. Among the debiasing GNN frameworks, CAL performs the best, being unaffected by changes in the bias degree for this variant of the Spurious-Motif dataset. However, the performance of UACG, while not bad, requires further improvement. On the other hand, UACG outperforms other models significantly on the CMINST-75sp dataset, exhibiting a considerable performance gap. Nevertheless, there is a different phenomenon in DIR. Although its accuracy is the worst, its accuracy does not decrease with the increase of bias degree, and it seems most stable. Further verification is in section 5.8..

5.5 Performance On Real-world Datasets (RQ2)

To evaluate UACG's performance on real-world datasets, we conducted experiments on COLLAB and Graph-SST2 datasets in this section. The experimental results are summarized in Table 5.5.

Setup. The COLLAB dataset consists of pure cooperative network data, without node features.

Model	Graph-SST2	COLLAB
GCN	$80.337_{\pm 0.46}$	$70.583_{\pm 2.05}$
GIN	$80.366_{\pm 0.24}$	$68.179_{\pm 5.22}$
GAT	$\mathbf{81.62_{\pm 0.3}}$	$71.29_{\pm 1.58}$
DIR	$81.178_{\pm 4.38}$	$56.501_{\pm 1.8}$
CAL	$80.732_{\pm 1.86}$	$70.359_{\pm 3.09}$
UACG	$81.504_{\pm 0.83}$	$\mathbf{72.49_{\pm 0.62}}$

Table 5.5: The ACC(%) of graph classification for real-world datasets. The best results are highlighted in bold.

To address this issue, we used eigenvector centrality to initialize the features of all nodes. Eigenvector centrality measures the influence of a node in a network by assigning relative scores based on connections to high-scoring nodes. The first three models in Table 5.5 are basic GNN models without a debias framework, while the last three models incorporate a debias framework.
Results. Analyzing Table 5.5, we can observe that GAT achieves the best overall performance for graph label predictions on the Graph-SST2 dataset, with an accuracy of 81.62%. Among the debias frameworks, UACG performs the best. Although the **ACC** results of GAT and UACG are similar, GAT exhibits smaller variance in its five results, indicating greater stability in this dataset. From the third column of Table 5.5, we can see that UACG outperforms other methods and demonstrates higher stability compared to other models. However, DIR's performance is not satisfactory, suggesting vulnerability to distribution shifts in certain datasets.

5.6 Ablation Studies (RQ3)

In this section, we investigate the impact of each main component and different uncertainty estimation methods on both synthetic and real-world datasets. The experimental results are summarized in Figure 5.3.
Setup. As mentioned in section 4.4, we consider two uncertainty subgraphs sampling methods. The first method involves sampling the top 20% nodes with the highest attention scores to generate the uncertainty subgraph. The second method uses Locality Sensitive Hashing (LSH) to categorize nodes' representations into six buckets and randomly sampls one node from each bucket.
Results. In Figure 5.3(a), when using the TopK uncertainty subgraph sampling method on the CMINST-75sp 0.8 dataset, the full model achieves an accuracy of 74.89%. Removing the label conformity (LC) component results in a drop in accuracy to 62.3%, indicating its importance in capturing local information. The removal of the uncertainty-aware classification (UC) component results in an accuracy of 65%, suggesting its significant contribution. The

	CMINST-75sp	Models					
Metric	Data Bias Degree	GIN	GCN	GAT	CAL	DIR	UACG
TPR	0.8	-	-	-	23.214	28.686	**55.159**
PPV		-	-	-	19.436	24.017	**46.18**
TNR		-	-	-	81.055	82.133	**87.344**
NPV		-	-	-	84.281	85.401	**90.820**
TPR	0.9	-	-	-	25.907	20.53	**29.90**
PPV		-	-	-	21.691	17.188	**25.03**
TNR		-	-	-	81.586	80.527	**82.37**
NPV		-	-	-	84.83	83.731	**85.651**
TPR	0.95	-	-	-	25.945	32	**43.833**
PPV		-	-	-	21.72	26.796	**36.699**
TNR		-	-	-	81.593	82.78	**85.11**
NPV		-	-	-	84.840	86.08	**88.502**

Table 5.6: The evaluation of the captured causal and non-causal subgraphs based on True Positive Rate (TPR), Positive Predictive Value (PPV), True Negative Rate (TNR), and Negative Predictive Value (NPV) is presented in Table 5.6. The best results are highlighted in bold. "-" represents cases where the debias framework cannot be employed, resulting in the absence of a causal subgraph. Consequently, it becomes impossible to conduct a quality analysis on it.

removal of both LC and UC causes a significant decline in accuracy to 53.07%, indicating their combined effect. For the COLLAB dataset, the full model achieves an accuracy of 72.26%. Removing LC decreases accuracy to 62.4%, while removing UC results in an accuracy of 68.8%. The removal of both components leads to an accuracy of 60.8%. Therefore, employing the Top-K uncertainty subgraph sampling method reveals that LC plays a more crucial role, as its exclusion causes the largest decline in accuracy. Moreover, removing both components leads to a maximum accuracy decline of 21.82%. In Figure 5.3(b), when using the LSH uncertainty subgraph selection method, the impact of the components is similar, except for the COLLAB dataset. For COLLAB, the full model achieves an accuracy of 74.26%. Removing the LC component leads to a slightly lower accuracy of 71.46%, while removing the UC component results in an accuracy of 69.33%. Removing both LC and UC together leads to an accuracy of 68.13%. The impact of UC is greater than that of LC. Overall, better results can be obtained with LSH on the COLLAB dataset, while TopK performs better on CMINST-75sp.

5.7 Disengagement Quality (RQ4)

Accurately identifying the causal and non-causal components during training is essential for effectively debiasing graph encoders. In this section, we examine the quality of the causal and non-causal parts discovered by all the methods.

Setup. In this section, we consider the causal nodes as positive values and the non-causal nodes as negative values for better understanding and explanation. We evaluate the performance based on the CMINST-75sp dataset, which we generated following the details of the study [2]. This allows us to obtain a deeper understanding of the dataset and identify which nodes belong to the ground truth causal and non-causal parts. As more than 80% of the nodes in this dataset belong to the non-causal part, the false negative values captured by models are very small. Consequently, the precision value is very similar to the F1 score, and we do not introduce the F1 score in our evaluation. Additionally, the regular GNN baseline models do not generate causal or non-causal subgraphs during training or testing, as they aggregate all neighbors' information without distinguishing between causal and non-causal parts.

Results. The results are summarized in Table 5.6. Based on this table, we make the following observations: UACG consistently produces high-quality causal and non-causal subgraphs. Apart from UACG, the quality of subgraphs found by DIR is generally better than CAL. However, when the data bias degree reaches 0.9, CAL performs better than DIR. As more than 80% of the nodes belong to the non-causal part in this dataset, the overall difference in non-causal evaluation (NPV and TNR) among the three models is not significant. The most significant difference is observed at a data bias degree of 0.8, with an NPV difference between CAL and UACG, but it is still less than 7%. On the other hand, UACG shows a significant improvement in identifying causal nodes, demonstrating its superior resolution ability.

5.8 Visualization Of Rationale (RQ5)

Setup. To intuitively understand the quality of the rationale, we analyse several cases of generated rationale by UACG against strong baselines for CMINST-75sp. The five cases correspond to the numbers: **0 , 1, 3, 5, 7**. In Figure 5.4, (a) shows the original images of these numbers, and (b) shows the converted graphs with ground truth causal rationale nodes. Figures 5.5 to 5.7 depict the rationale visualizations captured by the three GNN debias frameworks under different data bias degrees: **0.8 , 0.9, 0.95**.

Results Firstly, comparing the ground truth causal rationale graph in Figure 5.4(b), we can observe that UACG can remarkably identify causal rationale nodes when the data bias is 0.8, as shown in Figure 5.5(a). Although it is not possible to directly see the numbers in each output result, UACG consistently captures the correct rationale nodes. In contrast, CAL in Figure 5.5(b) and DIR in Figure 5.5(c) tend to capture background color nodes as causal nodes. Secondly, when the data bias is 0.9, UACG captures some of the ground truth causal rationale nodes but is

noticeably affected by non-causal nodes, as shown in Figure 5.6(a). However, the performance of DIR has significantly improved, as seen in Figure 5.6(c). Finally, in Figure 5.7(a), UACG again demonstrates excellent visual results. Although UACG generally outperforms CAL and DIR, the results from DIR appear more stable across different data bias degrees. However, CAL consistently introduces non-causal nodes to aid model predictions. These visualizations provide insights into the performance and behavior of the GNN debias frameworks, highlighting the strengths and weaknesses of each method.

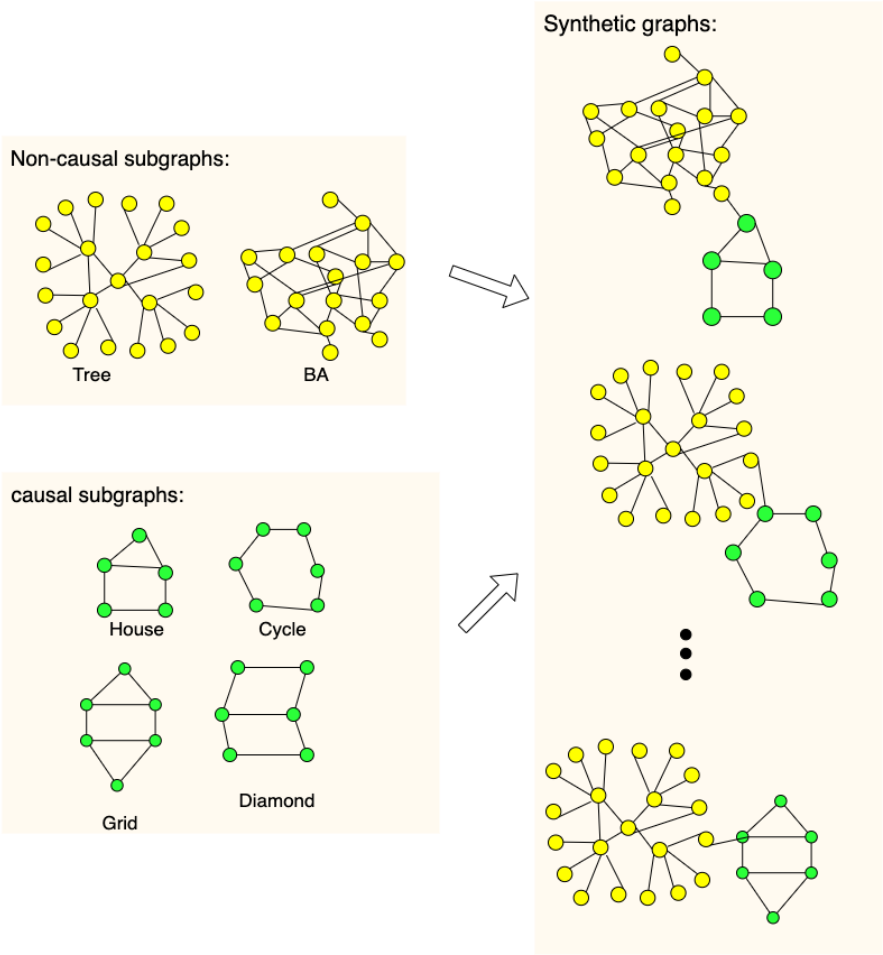


Figure 5.1: Spurious-Motif Dataset.

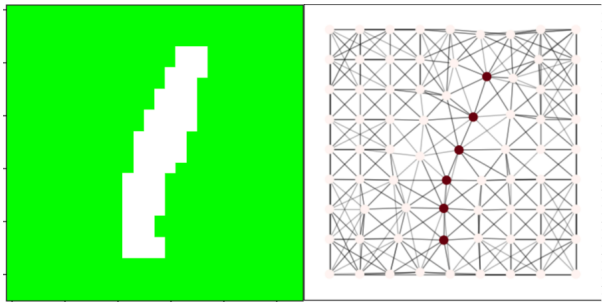
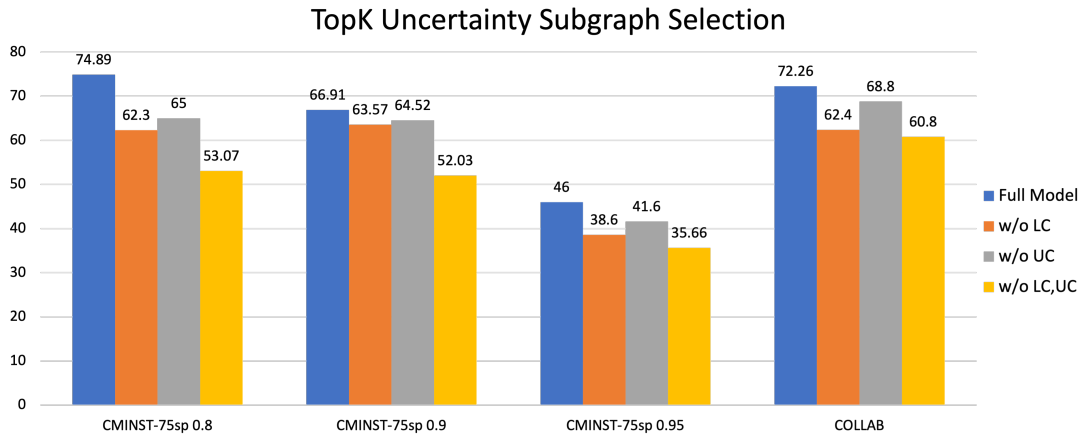
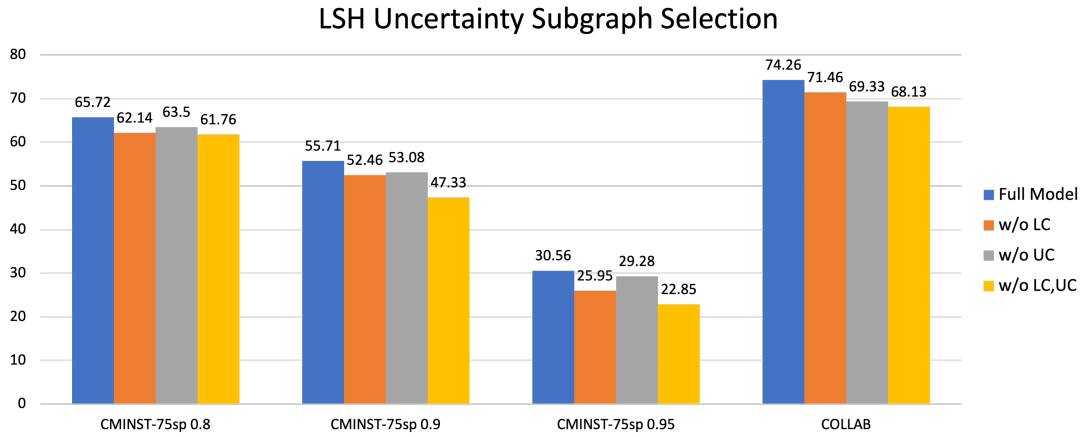


Figure 5.2: CMNIST-75sp Dataset.



(a) Top-K Uncertainty Subgraph Selection



(b) LSH Uncertainty Subgraph Selection

Figure 5.3: The comparison of different components in UACG with ACC(%). LC represents the label conformity component. UC represents the uncertainty-aware classification.

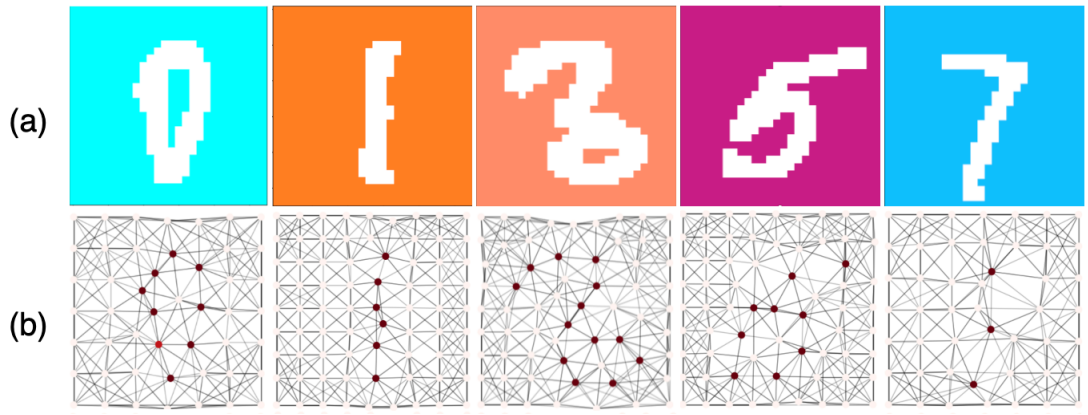


Figure 5.4: Original Image and Ground Truth Graph.

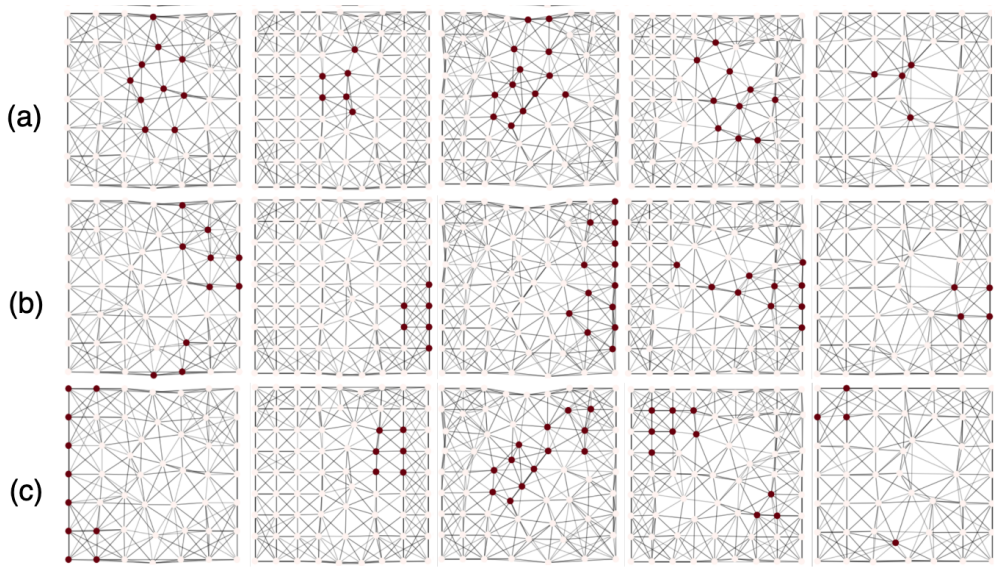


Figure 5.5: The causal graph captured by three GNN debias framework with bias degree of 0.8. (a): Causal graphs generated by UACG.(b): Causal graphs generated by CAL. (c): Causal graphs generated by DIR.

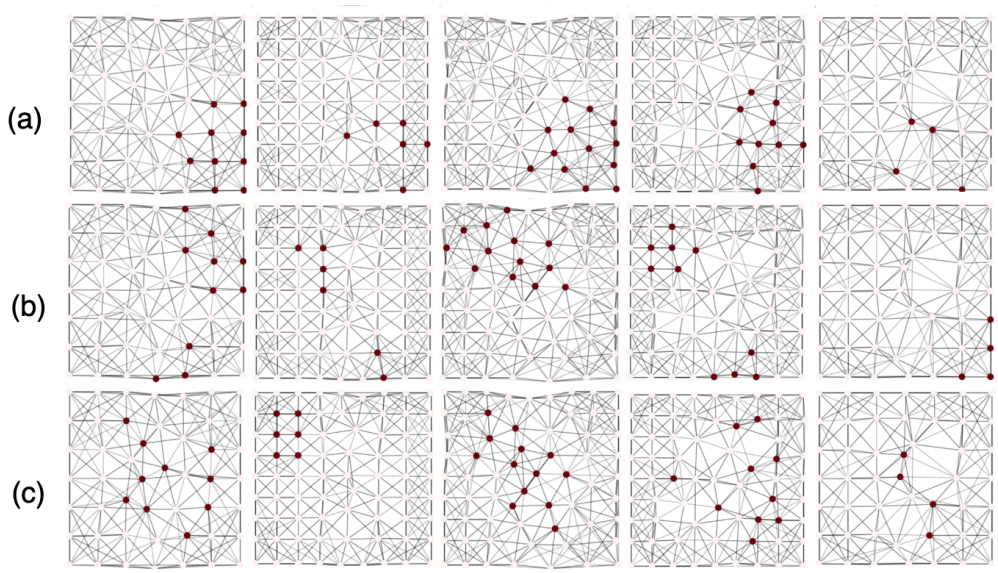


Figure 5.6: The causal graph captured by three GNN debias framework with bias degree of 0.9.(a): Causal graphs generated by UACG.(b): Causal graphs generated by CAL. (c): Causal graphs generated by DIR.

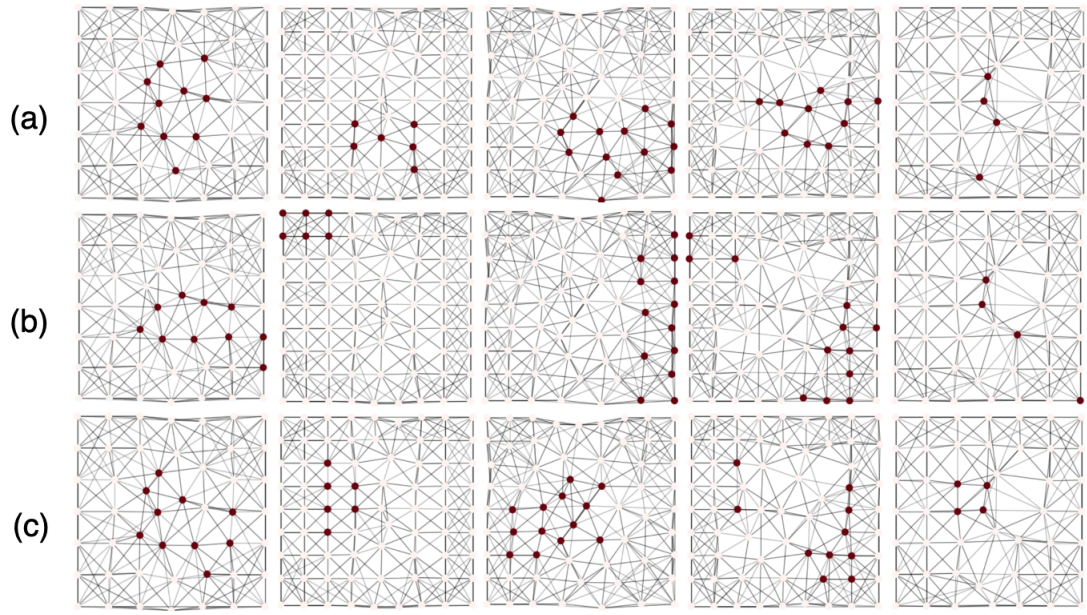


Figure 5.7: The causal graph captured by three GNN debias framework with bias degree of 0.95.(a): Causal graphs generated by UACG.(b): Causal graphs generated by CAL. (c): Causal graphs generated by DIR.

Chapter 6

Conclusions

The contributions of the study, as well as its limitations and recommendations for further research, are presented in the following chapter. Section 6.1 concludes the most important achievements of our study. Section 6.2 highlights the main limitations, and Section 6.3 presents the possible work for further research.

6.1 Achievement

The major achievements of our study are highlighted as follows.

- The accuracy of label prediction in most datasets exceeds most baseline models.
- The quality of the causal graph was finally found to outperform all baseline models.
- Utilizing the graphical method enables us to intuitively understand the process of removing bias, thereby enhancing the model's comprehension.

6.2 Limitations

In spite of current progress, there exist shortcomings to be tackled. The following details the limitations of our study.

6.2.1 Effectiveness of Capturing Small Graphs

UACG does not perform well on tiny graph data. From the overall framework of UACG, it is evident that it involves disentangling operations at least twice. Therefore, if a dataset contains tiny graphs, such as the Graph-SST2 dataset discussed in Section 5.5 for sentence sentiment analysis, where some sentences consist of no more than five words, errors may occur due to the

removal of all edges. We believe this could be the reason why UACG fails to surpass GAT on the Graph-SST2 dataset.

6.2.2 Hyperparameters Tuning

UACG includes a hyper-parameter K for Top-K uncertainty subgraph selection. In this study, we set K equal to the number of nodes multiplied by 20%. Similarly, as discussed in Section 5.4, the performance of UACG is not satisfactory on the Spurious-Motif dataset. Upon observing the structural characteristics of the data, we can determine that each graph size is at least 200 nodes, and the ground truth causal node count is only six. In this scenario, a 20% selection is too large for this dataset. Due to time constraints, we have not addressed the issue of parameter optimization.

6.2.3 Time Costs

The current framework involves at least two disentanglement processes. While it leads to improvements in accuracy and quality, the training time is also doubled compared to DIR. Simplifying the model framework remains a worthwhile goal for further optimization.

6.3 Future Work

The followings list the suggestions for future work.

6.3.1 Improve All The Limitations Mentioned Above

The top priorities for further improvement are addressing the three limitations mentioned earlier. The first limitation is particularly significant as it restricts the applicability of our model to datasets containing tiny graphs, where it is not feasible to fully implement all the steps in the framework. Therefore, in future work, we will focus on optimizing the model architecture and exploring the best parameters to make the model as effective as possible.

6.3.2 Subgraph Generation

As mentioned earlier, we generated subgraphs based on the L-hop algorithm. However, we are curious if there are other algorithms for subgraph generation that can assist in disentangling the causal and non-causal parts. In future work, we will further explore this technological aspect to optimize the architecture.

6.3.3 Graph Encoder Complexity

The current framework heavily relies on obtaining attention scores for nodes and edges to optimize the model during training. In the future, we will conduct further experiments and explore methods that can reduce the complexity of the existing architectures without losing accuracy and quality.

Bibliography

[1] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems*, 32, 2019.

[2] Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning de-biased representations with biased representations. In *International Conference on Machine Learning*, pages 528–539, 2020.

[3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[4] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*, pages 874–883, 2020.

[5] Michael Bronstein. Beyond message passing: a physics-inspired paradigm for graph neural networks. *The Gradient*, 2022.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. 2014.

[7] Daniele Calandriello, Alessandro Lazaric, Ioannis Koutis, and Michal Valko. Improved large-scale graph learning through ridge spectral sparsification. In *International Conference on Machine Learning*, volume 80, pages 688–697, 2018.

[8] Shiyu Chang, Yang Zhang, Mo Yu, and Tommi Jaakkola. Invariant rationalization. In *International Conference on Machine Learning*, pages 1448–1458, 2020.

[9] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020.

[10] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

[11] Jia Cui, Brian Kingsbury, Bhuvana Ramabhadran, Abhinav Sethy, Kartik Audhkhasi, Xiaodong Cui, Ellen Kislal, Lidia Mangu, Markus Nussbaum-Thom, Michael Picheny, et al. Multilingual representations for low resource speech recognition and keyword search. In *2015 IEEE workshop on automatic speech recognition and understanding*, pages 259–266, 2015.

[12] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pages 417–426, 2019.

[13] Jiarui Feng, Yixin Chen, Fuhai Li, Anindya Sarkar, and Muhan Zhang. How powerful are k-hop message passing graph neural networks. In *Advances in Neural Information Processing Systems*, 2022.

[14] Zhan Gao and Elvin Isufi. Learning stable graph neural networks via spectral regularization. *arXiv preprint arXiv:2211.06966*, 2022.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272, 2017.

[16] Siavash Golkar, Micheal Kagan, and Kyunghyun Cho. Continual learning via neural pruning. In *Real Neurons & Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence*, 2019.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. volume 30, 2017.

[18] Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. An attention-based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4132–4139, 2020.

[19] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference*, pages 2704–2710, 2020.

[20] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. Continual learning with node-importance based adaptive group sparse regularization. volume 33, pages 3647–3658, 2020.

[21] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*, 2021.

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[23] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison Wesley, 2006.

[24] Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[25] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5:1–42, 2020.

[26] John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. Graph convolutional networks with motif-based attention. In *international conference on information and knowledge management*, pages 499–508, 2019.

[27] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743, 2019.

[28] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *International conference on machine learning*, pages 3734–3743, 2019.

[29] Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, 2016.

[30] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. page 97–109, 2019.

[31] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[32] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[33] Zenan Li, Qitian Wu, Fan Nie, and Junchi Yan. Graphde: A generative framework for debiased learning and out-of-distribution detection on graphs. *Advances in Neural Information Processing Systems*, 35:30277–30290, 2022.

[34] Wanyu Lin, Hao Lan, and Baochun Li. Generative causal explanations for graph neural networks. In *International Conference on Machine Learning*, pages 6666–6679, 2021.

[35] Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *International Conference on Machine Learning*, pages 6837–6849, 2021.

[36] Yang Liu, Xiang Ao, Fuli Feng, and Qing He. Ud-gnn: Uncertainty-aware debiased training on semi-homophilous graphs. In *Conference on Knowledge Discovery and Data Mining*, pages 1131–1140, 2022.

[37] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems)*, 33:19620–19631, 2020.

[38] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *Workshop on Graph Representation Learning and Beyond*, 2020.

[39] Judea Pearl. *Causality: Models, Reasoning and Inference*. New York: Cambridge University Press, 2000.

[40] Judea Pearl. Interpretation and identification of causal mediation. *Psychological methods*, 19(4):459, 2014.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[42] Jan Schuchardt, Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. Collective robustness certificates. In *International Conference on Learning Representations*, pages 4–7, 2021.

[43] Yanhu Mo Chuan Shi Jian Tang Shaohua Fan, Xiao Wang. Debiasing graph neural networks via learning disentangled causal substructure. In *Advances in neural information processing systems*, 2022.

[44] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30:83–98, 2013.

[45] Daniil Sorokin and Iryna Gurevych. Modeling semantics with gated graph neural networks for knowledge base question answering. In *International Conference on Computational Linguistics*, pages 3306–3317, 2018.

[46] Yongduo Sui, Xiang Wang, Jiancan Wu, Min Lin, Xiangnan He, and Tat-Seng Chua. Causal attention for interpretable and generalizable graph classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1696–1705, 2022.

[47] Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. In *International Conference on Learning Representations*, 2018.

[48] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[49] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[50] Xiaoqi Wang and Han Wei Shen. GNNInterpreter: A probabilistic generative model-level explanation for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[51] Ying-Xin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat seng Chua. Discovering invariant rationales for graph neural networks. In *International Conference on Learning Representations*, 2022.

[52] Yonghui Wu, Jeremy L Warner, Liwei Wang, Min Jiang, Jun Xu, Qingxia Chen, Hui Nian, Qi Dai, Xianglin Du, Ping Yang, et al. Discovery of noncancer drug effects on survival in electronic health records of patients with cancer: a new paradigm for drug repurposing. *JCO clinical cancer informatics*, 3:1–9, 2019.

[53] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[54] Sibei Yang, Guanbin Li, and Yizhou Yu. Dynamic graph attention for referring expression comprehension. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4644–4653, 2019.

[55] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[56] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. volume 31, 2018.

[57] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[58] Muhammad Rehman Zafar and Naimul Khan. Deterministic local interpretable model-agnostic explanations for stable explainability. *Machine Learning and Knowledge Extraction*, 3:525–541, 2021.

[59] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 793–803, 2019.

[60] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[61] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Exploring edge disentanglement for node classification. In *Proceedings of the ACM Web Conference*, pages 1028–1036, 2022.